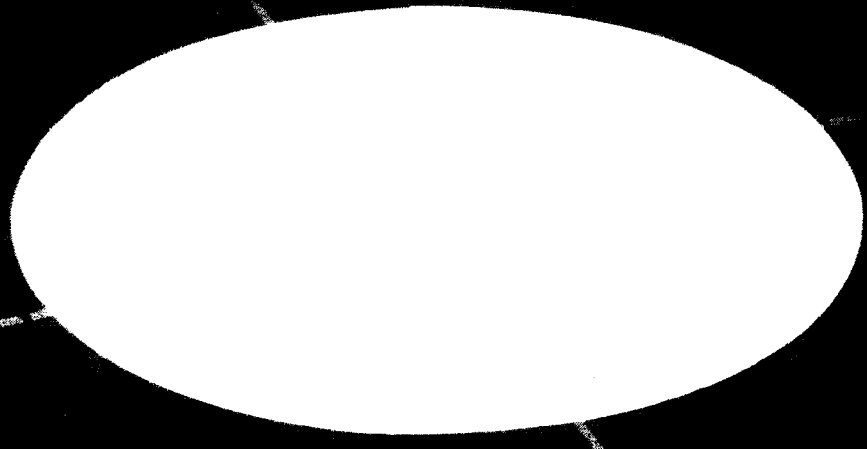
N64 11119
CODE-1
NASA CR-52688

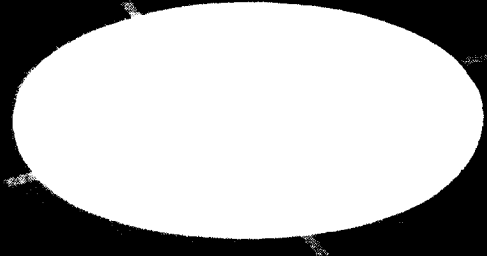
ENGINEERING DESIGN CENTER



OTS PRICE

XEROX $ 12.50 ℓ

MICROFILM $ 5.12 ℓ



1716000
Corp. author:
Case Inst. of Tech.,
Cleveland, Ohio
@ Digital Systems Lab.

N64 11119*
CODE-1
(NASA CR-52688;
EDC-1-63-16)
OTS: $12.50 ph, $5.42 mf

This Research Was Sponsored by

THE NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

(2)

Analysis and Synthesis of Transition-Coupled Asynchronous Counters

Report No. EDC 1-63-16

by

Walter Arnstein and Harry W. Mergler

Harry W. Mergler
Professor of Engineering
Principal Investigator
(NASA Grant NsG-36-60)

Digital
Systems
Laboratory

October 1963

174 p refs

ABSTRACT

11119

The transition-coupled asynchronous counter is a single-input sequential circuit containing interstage pulse coupling in the form of transition signals, produced by the memory units in the process of switching.

In this study, a logic algebra is introduced for describing the action of a transition-coupled circuit having roughly uniform interstage delays and comparatively small delays due to combinational circuitry. Called subinterval logic, it envisions all action in the circuit as taking place at a finite set of discrete event times.

A matrix method for deriving loop equations for transitions in a transition-coupled circuit is described and a test for inherent stability of the circuit is formulated.

A simple matrix technique for the analysis of a stable transition-coupled counter is derived. It includes a test for input critical races.

A standard-algebraic method for the synthesis of transition-coupled counters is described. This method uses the standard binary adder as a model for the synthesis.

All techniques are illustrated with examples.

Extensions of subinterval logic to the multi-input transition-coupled circuit are outlined and recommendations for additional study are made.

AUTHOR

ACKNOWLEDGMENT

The author wishes to express his deepest appreciation to Professor Harry W. Mergler, Director of the Digital Systems Laboratory of the Case Engineering Design Center, for suggesting this challenging problem and for his encouragement and helpful advice in the development of this paper. The author is also indebted to the National Aeronautics and Space Administration for sponsoring the research from which this report evolved.

The invaluable help of Miss Diane Ross, who advised the author in the preparation of the manuscript and typed the preliminary and final drafts, is most gratefully acknowledged.

TABLE OF CONTENTS

KEY TO SYMBOLS AND NOTATION

General Symbols

$(\underline{A})$	in subinterval analysis, the underlined portion of a mixed product of literals. E.g., $\underline{ABC}$ in DB$\underline{ABC}$.
A	an internal state variable
$(A_k B_k C_k \ldots.)$	the k^{th} word in a general counting sequence
a	(without subscript) see b
a_k	k^{th} addend digit for a binary adder, corresponding to an addition of $a_k 2^k$
B	an internal state variable
b	in counter synthesis, an arbitrary constant corresponding to v for a redundant combination
C	an internal state variable
$[C]^t$	connection matrix for a sequential circuit
$[C]_r^t$	reduced connection matrix for a sequential circuit
c	see b
D	an internal state variable
d	see b
$[E]$	in counter synthesis, the matrix relating the v coefficients to the w coefficients
e	see b
F_k	at event time kt, a level function of the last stable state of the circuit
f	see b; also, a Boolean function

f_k	a general pulse signal at event time kt
g	a Boolean function
$[I]$	the identity matrix
I_k	time interval within which a circuit switches from stable state $[Q^{k-1}]$ to stable state $[Q^k]$
i	a running index
j	a running index
k	a running index
$\{L\}$	set of level signals defined at all times
Mt	the last event time in a transition between stable states. The circuit is stable following this event time.
M_{max}	the number of events (not counting the event at Ot) comprising the longest transition between two successive stable states for a counter.
m	a running index
m_j	the j^{th} minterm
N	in recursive transition equations, the number of subintervals required for a signal to traverse the longest closed proper path in the circuit
n	number of outputs for a sequential circuit. Also, general expression for the number of stages in a counter
$[O]$	the zero matrix
Ot	event time at which command pulse p is applied to a circuit at rest and all circuit action begins

$\{P\}$	set of pulse signals defined at all times
p	transition command pulse which initiates all circuit action
$[Q]$	internal state vector
$[Q']$	next internal state vector
$\bar{Q}$	not Q
$\bar{Q}'$	not Q'
Q	a general state variable
Q'	the next state of Q
q	a pulsed variable
R	a state variable
R_Q	reset input to memory unit Q
r	a pulsed variable; also, (as an index) the number of inputs to a sequential circuit
r	in counter synthesis, the number of redundant states for a particular counter
S_Q	set input to memory unit Q
s	number of internal states for a sequential circuit
$\{s_i\}$	set of subintervals over all intervals I^k whose end points are event times (i-1)t and (i)t, respectively
T_Q	trigger input to memory unit Q
$[T]$	transition vector for a sequential circuit
$[T_0]$	zeroth transition vector for a sequential circuit

T_k	kth transition vector for a sequential circuit
$[\mathcal{T}_k]$	the kth transition matrix, whose component columns are different transition vectors $[T_k]$
t	interstage time delay; the length of the logic subinterval, defined as the longest interstage delay in a circuit. As a superscript, denotes reference to one subinterval ago. The superscript kt (k an integer) denotes reference to k subintervals ago.
u	a pulsed signal
V	general numerical value of the binary number encoding of the present internal state in a counter
V'	general numerical value of the binary number encoding of the next stable internal state in a counter
V_k	numerical value of the binary number encoding of the kth word in a count sequence
V'_k	numerical value of the binary number encoding of the $(k+1)^{th}$ word in a count sequence
v	general numerical value of the difference in the binary number encodings of two successive stable states of a counter. Also, a pulsed signal.
$[v]$	column vector of v coefficients
v_k	the numerical change in the binary number encoding of a counter state when switching from its k^{th} state to its $(k+1)^{th}$ state
w	a pulsed signal
$[w]$	column vector of w coefficients

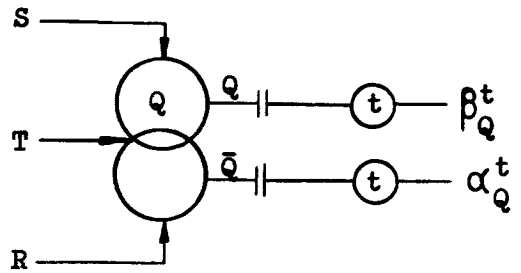
w_k	in counter synthesis, the coefficient of the k^{th} exterm in the weighted exterm expansion for v
$[X]$	input vector for a sequential circuit
x	a product term of literals
y	a product term of literals
$[Z]$	output vector for a sequential circuit
z	a product term of literals

Special Symbols

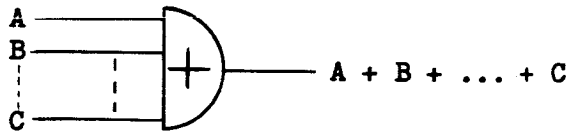
α_Q	Boolean variable denoting the transition of Q from Q = 0 to Q = 1. Also, the pulse signal emitted by stage Q in performing this transition obtained by differentiating the level signal $\bar{Q}$.
$\{\alpha_i\}$	set of pulse signals produced by 0 - 1 transitions of memory elements Q_i
β_Q	Boolean variable denoting the transition of Q from Q = 1 to Q = 0. Also, the pulse signal emitted by stage Q in performing this transition, obtained by differentiating the level signal Q.
$\{\beta_i\}$	set of pulse signals produced by 1 - 0 transitions of memory elements Q_i
Δ_Q	transition of either polarity on the part of stage Q
δ	general symbol for the time interval preceding an event time, during which all pulsed signals arrive at their destinations.
δ_j	the length of the interval preceding event time jt during which all pulsed quantities reach their destinations.

$\in$	connective indicating membership in a set. Read "is a member of ....."
$\notin$	the converse of $\in$. Read "is not a member of ....."
$\left[\Pi_C\right]^{kt}$	product of transition matrices $\left[C\right]^t$ through C^{kt}, with diagonal terms not used in multiplication
$\left[\Pi_C\right]_r^{kt}$	reduced product of transition matrices
$\sum_{j=A}^{B}$	multiple ring sum (exclusive OR) over j from j = A to j = B inclusive; also, multiple algebraic sum (PLUS)
τ_{max}	the longest level signal rise time in a sequential circuit
$\emptyset$	0 or 1 - - a "don't care" Boolean quantity
$\bigcup_{j=A}^{B}$	multiple logical union (OR) over j, from j = A to j = B inclusive
$\oplus$	ring sum (exclusive OR) connective
$\triangleq$	definition connective, read as "is defined as ....."
+	the logic union (OR) connective. In algebraic synthesis technique, the addition (PLUS) connective
*	the concatenation connective of regular expression algebra
=	identity connective

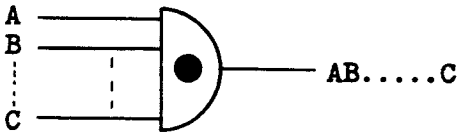
Logic Symbols



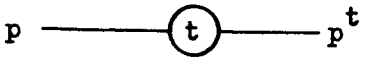
RST memory unit
(flip flop)



OR gate



AND gate



pulse delay
(1 subinterval)

I. INTRODUCTION

The sequential circuit comprises an important part of any digital computer or automatic control system. In its engineering form it consists basically of an aggregate of memory units having combinational circuitry at their inputs and outputs, as shown in the functional block diagram in Figure 1.1. The outputs are functions of the inputs and the states of the memory units. Every time the input configuration changes, it defines a new set of states for the memory units and a new output configuration.

In formal terms, the sequential circuit is a structure characterized by an input vector $[X]$, an internal state vector $[Q]$, and an output vector $[Z]$. Functionally, these entities are related by equations of the form

$$Z_i = f_i(X_j, Q_k) \qquad (1.1)$$

$$Q'_m = g_m(X_j, Q_k) \qquad (1.2)$$

$$0<i\leq n-1$$

$$0<j\leq r-1$$

$$0<k,m\leq s-1$$

where $[Q']$ is the next internal state vector.

The structure described above will be recognized as characterizing all arithmetic circuitry and most control circuitry in

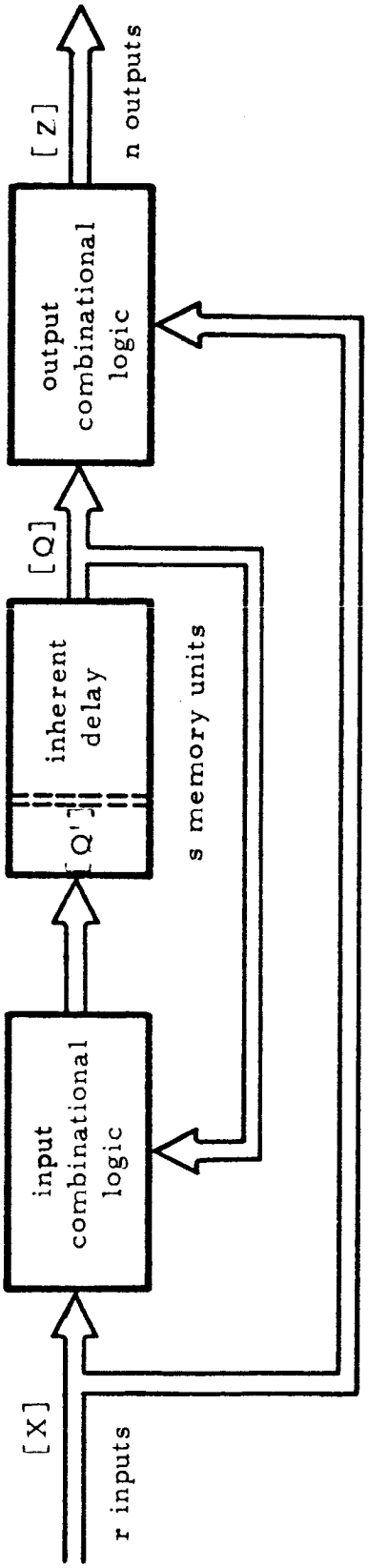

[X]
r inputs
input combinational logic
[Q']
s memory units
inherent delay
[Q]
output combinational logic
[Z]
n outputs

Figure 1.1

digital control equipment and is thus a common design problem. One of the main tasks in its design is to determine the number of internal states the circuit is to have, for the unit is generally specified in terms of input-output relations with no direct reference to the internal state. The classic works in the field of state identification techniques and reduction of a circuit to a "minimal state machine" are by Moore [26], Mealy [23], and Huffman [16,17]. Among more recent contributors, Unger and Paull [32], Ginsburg [12,13], and Unger [31] also considered this problem at length. Netherwood [28] in addition introduced helpful manipulative techniques to aid in the reduction and considered the reduction of combinational circuitry for the minimal circuit as well.

If the internal states are not previously specified, or coded, the designer can choose an internal code to minimize cost according to some criterion, or provide for optimum reliability. Excellent treatments of this problem can be found in papers by Armstrong [1,2], Hartmanis [14], and Stearns and Hartmanis [30].

Once the size of the memory unit has been established and an internal code has been chosen for a sequential circuit, the design problem focuses on the memory unit itself. A typical synthesis problem, for example, might be to specify the input

connective circuitry and memory inputs when the relation between inputs and internal states is specified. The analysis problem, on the other hand, might involve the determination of the behavior of a sequential circuit of known structure in response to an input not previously used. In either case, the task can proceed only if a valid mathematical model of the actual memory unit has been developed. In addition, a convenient logic algebra should be available to render a clear-cut description of the operation of the memory unit. The former problem does not present much difficulty, since the mathematical model can be built up directly by experimental observation. In contrast, the more intangible problem of logic formulation is far from simple. For example, the ordinary Boolean algebraic means used to describe synchronous counters in the form

$$[Q'] = f([Q]) \qquad (1.3)$$

fails to give an adequate description of a large class of asynchronous systems, as will be shown in Chapter II. Overcoming this particular problem is part of the objective of this study.

The Counter

This paper will consider the analysis and synthesis of a particular type of sequential circuit, namely the single input

circuit, often called an operation counter, or more simply, a counter. This circuit changes its internal state whenever a transition command is applied to it, according to a specified logical relation. While a full description of the response of the system is not given, it is assumed that the counter eventually reaches its next internal state and remains there indefinitely unless a new transition command is applied. Such a state is called a stable state. Any other states the memory unit assumes temporarily in the process of transition between two stable states will be called an unstable state.

A considerable body of literature on counter design and analysis is in existence. However, it is for the most part restricted to the synchronous case. That is, a stage in the counter is assumed to go from its present stable state to the next in a single transition, initiated by the transition command.

Such an implementation is generally costly in terms of logic circuitry but very reliable, providing timing problems have been eliminated.

A more sophisticated form of counter is the asynchronous counter, which can accept an input pulse any time it is in a state of rest and proceed to its next stable state in a series of actions determined and timed entirely by the counter's own circuitry. A particularly useful class of such counters will be

treated in this study.

This paper will deal with the analysis and synthesis of a transition-coupled asynchronous counter having as pulse inputs (a) a transition command pulse p, which initiates the circuit action and (b) fed-back transition pulses from the memory units themselves, obtained by differentiation of voltage levels. Pure delays, as well as the usual combinational circuitry are, of course, also permitted in the circuit.

A functional diagram of the above circuit is shown in Figure 1.2.

Certain restrictions are readily apparent from Figure 1.2. The principal of these is that only pulse quantities are permitted as inputs to the memory units. In addition, only state levels are considered as true outputs. A close examination of the diagram will also disclose a third stipulation: Since any transition in the memory unit can produce new pulse inputs, hence new states (and possibly additional transitions and pulses), it is clear that a number of unstable states may be assumed by the memory unit before it finally settles in a stable state. Therefore, it must be stipulated that the brief appearance of unstable states will not impair the operation of whatever is connected to the circuit. This is most often the case in sampled-data and quantized-data systems where the counter would be read

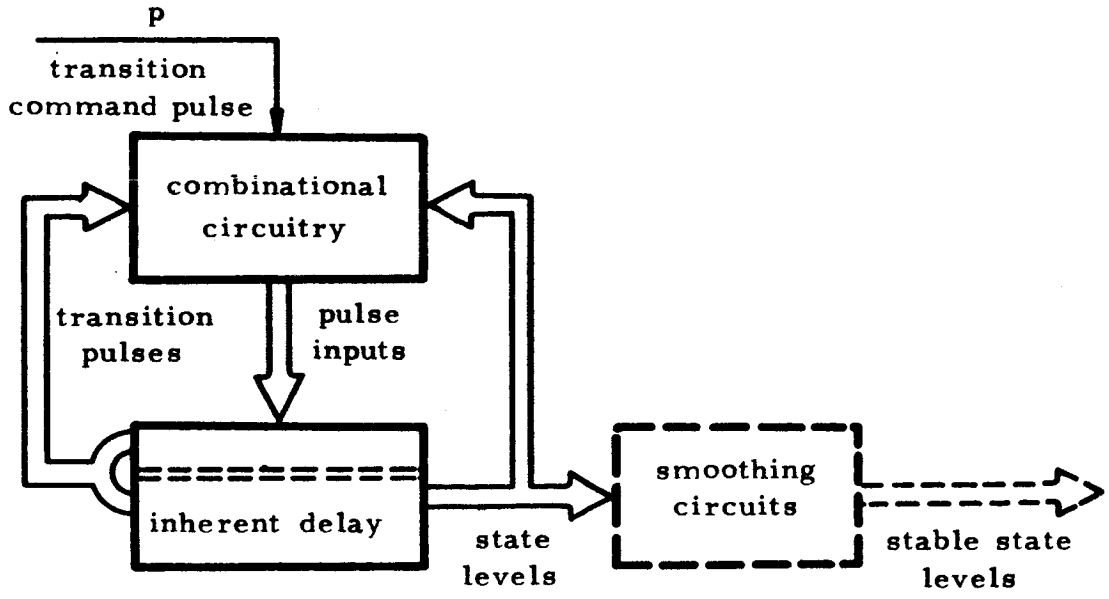

p
transition
command pulse
combinational
circuitry
transition
pulses
pulse
inputs
inherent delay
state
levels
smoothing
circuits
stable state
levels

Figure 1.2

only at discrete times not coincident with transitions.

To sum up the operation of a circuit based on the present mathematical model, the sequence of events in a typical "count" is:

1. The command pulse p is applied to the input line.
2. Entering the input combinational circuitry, the command pulse forms pulse inputs for the memory units. These inputs are in no way intentionally synchronized. Their times of application are determined by the natural delays within the input circuitry.
3. The individual memory units respond to their inputs by switching where applicable. In switching, they emit α or β (0 - 1 or 1 - 0, respectively) transition pulses.
4. The α or β pulses are fed back to the input combinational circuitry, possibly providing new pulse inputs for the memory units.
5. The process described in steps 3 and 4 continues until all feedback pulses cease and the circuit comes to rest. This defines the new stable state.

If a circuit fails to reach a stable state during any transition, it is inherently unstable. The analysis and design of such

circuits will not be considered in this study.

Considering the great complexity of the behavior of an asynchronous circuit as compared to that of an equivalent synchronous circuit, it becomes readily apparent that a concise logic applicable to the asynchronous circuit is seriously needed. Such a logic should be capable of describing any action, however peculiar, that an asynchronous circuit can take. Further, it should be mathematically sound and consistent. Finally, it should be reasonably simple to manipulate and should, of course, be capable of describing a synchronous circuit since the latter is merely a special case of the asynchronous circuit. Part of the objective of this paper is to develop such a logic. Supplementing rather than replacing the techniques presently used for synchronous analysis, the so-called subinterval logic permits known analysis techniques to be extended to include asynchronous sequential circuits having feedback pulse coupling.

A mathematical model for the type of memory unit to be considered in this paper will be described, along with the development of subinterval logic, in Chapter II. This chapter will also develop a technique for setting up circuit equations in subinterval logic and will show how the stability of a circuit is determined from these equations.

Chapter III discusses the analysis of a circuit once its

stability has been established. A rapid and compact method for performing the analysis is developed.

In Chapter IV the problem of synthesizing transition-coupled asynchronous circuits is considered. The techniques proposed are standard algebraic as well as logical.

Conclusions and recommendations for extensions and further investigation are presented in Chapter V.

II. SUBINTERVAL LOGIC

For purposes of mathematical representation, sequential circuits, whether synchronous or not, have for the most part been treated as synchronous, mainly because a well-developed mathematical procedure was available for this purpose. In considering internal states, the essence of this procedure was simply to express the next state variables of a circuit as Boolean functions of the input variables and the present states

$$Q'_m = g_m (X_i, Q_j) \qquad (2.1)$$

A further transformation was effected by applying the equations of state of the particular memory units in the circuit to the above equations. The result was a set of equations relating the general inputs for the memory units to the present states of the memories and any external variables. For pulse-input synchronous machines an implicitly assumed input is a clock pulse p without which the circuit could not operate.

The limited applicability of the preceding approach to the asynchronous switching circuit is readily apparent. First of all, it assumes that all logical operations prescribed for the circuit will be performed in a single step, keyed by the clock pulse; and secondly, it takes no cognizance of pulses and level changes as contrasted to level states. Thus, once the basic

character of a switching circuit was determined by analyzing it as a synchronous circuit, further simplification by design in terms of asynchronous logic had to be effected intuitively by the engineer. Needless to say, the ultimate simplicity of the final design was in a great measure dependent on the ingenuity of the individual engineer.

The use of differentiation logic, described by Mergler [24] and Marcus [21] helps place asynchronous analysis on a formal mathematical basis. Rather than studying the asynchronous circuit by considering its synchronous counterpart, differentiation logic introduces a new set of Boolean variables, $\{\alpha_i, \beta_i\}$, (notation after Mergler) representing pulsed quantities derived, respectively, from 0-1 and 1-0 transitions within the logical circuit. By logically considering transitions rather than final states, one can thus synthesize a large variety of sequential circuits directly, using leading-or trailing-edge logic, the inputs to all memory units being expressed in terms of existing states and transition pulses.

Before considering an example of synthesis using differentiation logic, it will be convenient to formulate a mathematical model for the type of memory unit to be used in this study. This unit is the pulse-input RST flip flop shown in Figure 2.1.

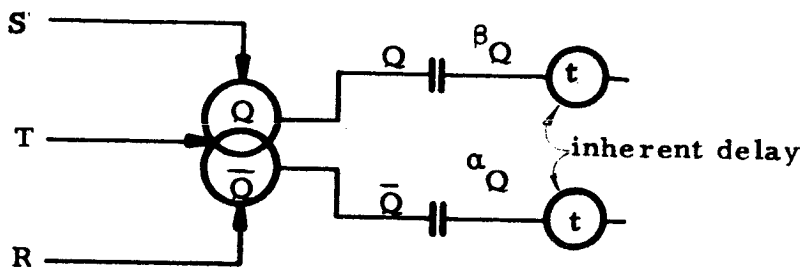


Figure 2.1

The unit has two level outpus, Q and $\bar{Q}$, one of which must be 1 and the other 0 in this simplest of models. In addition, there are two pulse outputs: α_Q, emitted when the unit switches from 0 to the 1 state (i.e. from Q = 0 to Q = 1); and β_Q, emitted when the unit switches from the 1 to the 0 state. The inputs R, S, and T are pulsed quantities assumed to be mutually disjoint at any given time. The relation among these quantities is:

$$\alpha_Q = (T + S)\,\bar{Q} \qquad (2.2)$$

$$\beta_Q = (T + R)\,Q \qquad (2.3)$$

$$Q' = S + T\,\bar{Q} + \bar{R}\bar{T}Q \qquad (2.4)$$

$$\bar{Q}' = R + TQ + \bar{S}\bar{T}\bar{Q} \qquad (2.5)$$

Also since R, S, and T are disjoint and the Change in Q is $\Delta_Q = \alpha_Q + \beta_Q = \alpha_Q \oplus \beta_Q$ (since $\alpha_Q \beta_Q = 0$), equations 2.4 and 2.5 may be written as

$$Q' = Q \oplus \Delta_Q = Q \oplus T \oplus S\bar{Q} \oplus RQ$$

$$= T \oplus \bar{R}Q \oplus S\bar{Q} \qquad (2.6)$$

$$\bar{Q}' = \bar{Q} + \Delta_Q = \bar{Q} \oplus T \oplus S\bar{Q} \oplus RQ$$

$$= T \oplus RQ \oplus \bar{S}\bar{Q} \qquad (2.7)$$

To demonstrate a case in which this logic proves very helpful, consider the design of a forward binary counter having stages A, B, C, and D corresponding to weights 8, 4, 2, and 1 respectively. By considering transitions of each stage in the process of counting,

one obtains

$$\begin{aligned} \alpha_D &= \bar{D}p \\ \beta_D &= Dp \\ \alpha_C &= \bar{C}Dp \\ \beta_C &= CDp \\ \alpha_B &= \bar{B}CDp \\ \beta_B &= BCDp \\ \alpha_A &= \bar{A}BCDp \\ \beta_A &= ABCDp \end{aligned} \tag{2.8}$$

This may be rewritten as

$$\begin{aligned} \alpha_D &= \bar{D}p \\ \beta_D &= Dp \\ \alpha_C &= \bar{C}\beta_D \\ \beta_C &= C\beta_D \\ \alpha_B &= \bar{B}\beta_C \\ \beta_B &= B\beta_C \\ \alpha_A &= \bar{A}\beta_B \\ \beta_A &= A\beta_B \end{aligned} \tag{2.9}$$

It is readily apparent that the simplest design for the counter is a set of T flip flops whose logic is given by

$$\begin{aligned} T_D &= p \\ T_C &= \beta_D \\ T_B &= \beta_C \\ T_A &= \beta_B \end{aligned} \qquad (2.10)$$

A hidden assumption in the familiar design shown above is that stage D switches first, thereby producing the switching signal for stage C, which in turn provides the switching pulse for stage B, etc. In short, interstage delays are assumed throughout the circuit. Yet, the algebra takes no account of these delays. Thus, it is only through sheer luck that this particular circuit functions as expected. As will be presently seen, the above method fails where additional delayed pulses from feedback paths appear at the flip flop inputs.

The basic shortcoming of ordinary differentiation logic is that it is formulated entirely on the assumption that during a single count, a particular memory unit goes from its initial state to its final state in one single transition, so that the actual operation of any memory unit can be determined fully by merely examining the final state and the initial state. In short, ordinary differentiation logic assumes that no more than one pulse, delayed or otherwise, is applied to a stage during any single count. Such is indeed the case for the ordinary binary counter. A general circuit with this characteristic is

called a normal fundamental mode circuit [11].

Many types of logical circuits, however, use feedback loops which are intended to provide certain stages with a series of input pulses. The stage in question thus assumes its final state by a series of switchings, emitting perhaps a number of important transition pulses in the process. Differentiation logic, in its original form, is incapable of furnishing a valid mathematical picture of this process. For example, consider the 2421 counter shown in Figure 2.2.

Applying differentiation logic directly yields

$$\begin{aligned} T_D &= p \\ T_C &= \beta_D + \alpha_A \\ T_B &= \beta_C + \alpha_A \\ T_A &= \beta_B \end{aligned} \qquad (2.11)$$

Note now that $T_C = 1$ whenever D switches from 1 to 0. The same is true of T_B whenever C switches from 1 to 0. This would seem to indicate that C and B will switch in such a case regardless of whether or not A ultimately switches from 0 to 1. Yet this is not a valid description of what happens in the actual circuit. A transition from 0111 to 1000 turns out to be temporary, as feedback from A sets stages B and C back to 1. Thus in a single count, stages B and C first switched from 1 to 0, emitting very necessary β pulses in the process, and then returned

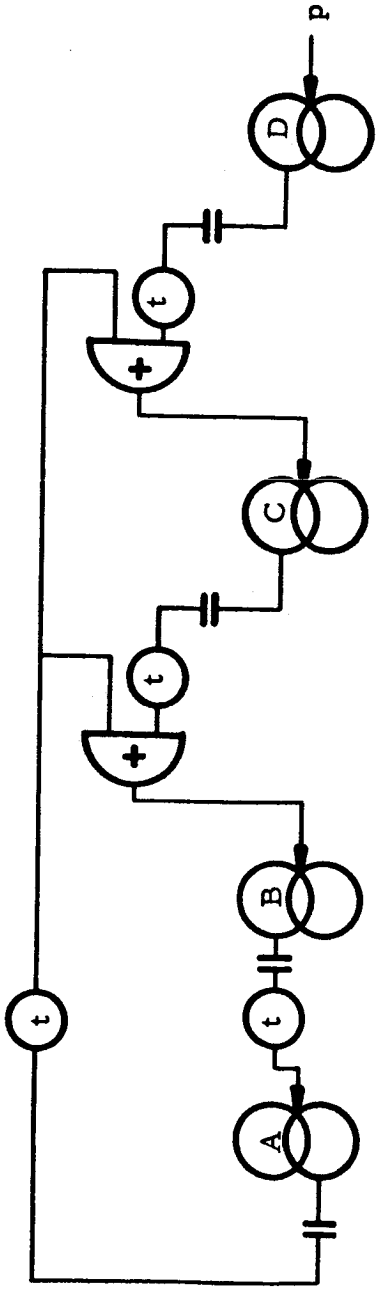


Figure 2.2

to their original states of 1. The effective T_B and T_C are thus 0 in terms of initial and final states and yet equal 1 as far as the emission of pulses is concerned. In short, ordinary differentiation logic fails when feedback loops are present in the logic because it contains no provision for the consideration of two or more input pulses appearing in a sequence, the results combining between two successive stable states.

The Logic Subinterval

Reflecting on the nature of ordinary differentiation logic, it may be said that its conceptual effect is to convert an asynchronous switching circuit to an equivalent synchronous unit by introducing new variables based on transitions of various outputs from one state to another. Its main shortcoming is that its basic synchronization interval is the fundamental clock interval itself; namely, only one transition between successive stable states on the part of any stage of a sequential circuit is permitted by the mathematics.

Once this handicap of ordinary differentiation logic is realized, an adaptation suggests itself readily. The interval between successive stable states is simply divided into a suitable set of subintervals. The precise length and number of these subintervals is not critical, but the number must be sufficient

to span the entire transition while the length must be sufficiently small to account for every significant event during the transition. A significant event here is intended to mean any event based on a delay and/or producing a delay without which the circuit could not operate. For example, in the case of the standard binary counter previously described, the significant events are the switchings of the individual memory units. Throughout this paper, only interstage delays associated with memory unit response time will be considered as significant. Delays encountered in gating will be considered as of a smaller order of magnitude and will not be depended on as design parameters. Furthermore, all interstage delays will be assumed to be approximately the same.

For purposes of the present discussion, divide the clock interval into a set of subintervals of length t, where t is the longest of the almost equal interstage delays in the circuit. The sequence of events when a transition command p is applied is shown in the timing diagram in Figure 2.3.

Now choose some time kt as a reference. The circuit is just responding to the k^{th} interstage delay and the inputs to the various flip flops are pulses originating at previous times, and now arriving due to delays. At this point, it will be convenient to introduce a notation to describe these delayed quantities. Let the superscript "t" indicate a delay of t,

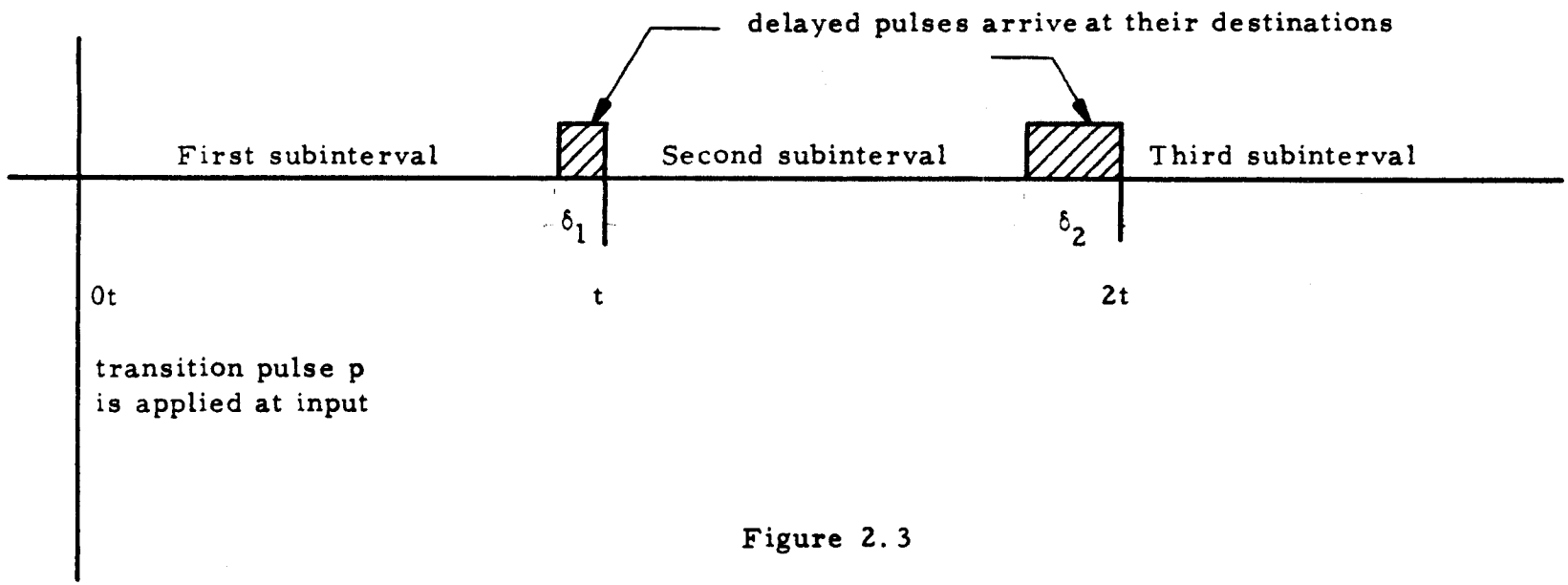

delayed pulses arrive at their destinations
First subinterval
Second subinterval
Third subinterval
δ_1
δ_2
0t
t
2t
transition pulse p
is applied at input

Figure 2.3

"2t" a delay of 2t, etc. Thus, if q is a pulsed quantity, q^t indicates that the pulse q originated one subinterval ago and is now arriving due to interstage delay. To cite a familiar entity, $\bar{Q}^t Q$ indicates Q was 0 one subinterval ago and is now 1. Hence, it had to switch one subinterval ago and an equivalent expression is thus α_Q^t.

At this point, a few basic postulates for subinterval logic must be stated. They are as follows:

1. Stability postulate: For every pair of successive states of a stable counter, $([Q^{k-1}], [Q^k])$, there exists a finite time interval I^k within which the transition from $[Q^{k-1}]$ to $[Q^k]$ takes place.
2. Every transition from state $[Q^{k-1}]$ to $[Q^k]$ takes place in a finite set of significant events that are approximately equally spaced in time.
3. Within every interval I^k, there exists a set of discrete event times, 0t, t, 2t, etc., such that all significant events can be observed within small increments δ_k preceding these times, where $\delta_k \ll t$.

4. There exists a set of subintervals s_i over all intervals I^k whose end points are event times (i-1)t and (i)t, respectively.
5. There exists a set of event times $\{0t\}$ over all I^k which mark the lower boundaries of the I^k. There also exist sets $\{t\}$, $\{2t\}$, etc. which mark the occurrences of significant events on all I^k.
6. On every I^k, there exists a set $\{L\}$ of level signals defined at all event times.
7. On every I^k, there exists a set $\{P\}$ of pulse signals defined at all event times.
8. For every I^k, there exists a command pulse $p \in \{P\}$ which marks event time 0t.
9. There exist subsets $\{\alpha_i\}$ and $\{\beta_i\}$ of $\{P\}$ produced by 0 - 1 and 1 - 0 transitions, respectively, of memory element Q_i.
10. Only pulsed quantities produce transitions in memory units and no level quantities are applied at their inputs.
11. At every event time, there exists a set $\{P^{mt}\} \in \{P\}$ of pulses originating m subintervals ago and now arriving due to delays.

12. At every event time, there exists a set $\{L^{mt}\} \in \{L\}$ of levels which were defined m subintervals ago.

13. Causality postulate: All circuit action starts with the command pulse p. Before p, the circuit is at rest in a stable state.

For the following postulates, assume lower case quantities are members of $\{P\}$ while capitalized quantities are members of $\{L\}$.

14. $[(q)^{nt}]^{mt} = q^{(m+n)t} \in \{P\}$

15. Asynchronism postulate:
$q^{kt} r^{kt} \notin \{P\}$

16. $(q^t)(A^t) = (qA)^t \in \{P\}$

17. $(A^t)(B^t) = (AB)^t \in \{L\}$

18. $(A^{nt} q^{mt})^{kt} = A^{(n+k)t} q^{(m+k)t} \in \{P\}$

Certain important theorems arise from these postulates.

Theorem 1: $p^{mt} p^{nt} = 0$ for $m \neq n$.

Proof: By the Causality Postulate (13) all circuit action started at a unique instant defined by the external application of p. At any event time, p^{mt} and p^{nt} refer to two different times, a contradiction of this postulate. Hence, $p^{mt} p^{nt} = 0$ for all $m \neq n$. Although expressions containing forms $p^{mt} p^{nt}$ will not appear as such in logical equations, this Theorem is important in establishing basic disjointness in such expressions as $A^{mt} p^{mt} + B^{nt} p^{nt}$.

Theorem 2: $p^{mt}A^{nt} = p^{mt}A^{mt}$ for $n \geq m$.

Proof: This again arises from the Causality Postulate. Since no action takes place before time Ot, the value of the level quantity A prior to Ot is its stable value still defined at Ot.

The next theorem is perhaps the most important theorem in subinterval logic. Before considering it, the following lemma must be established.

Lemma: At an event time kt, any signal $f_k \in \{P\}$ can be expressed in the form $F_k p^{kt}$, where $F_k \in \{L\}$ is a function of the state of the circuit at Ot.

Proof: Referring again to the Causality Postulate, any pulsed quantity had to originate with p, k subintervals ago. Hence, the presence of p^{kt} in f_k. Now since no other pulse could have originated at Ot and if it had, its use with p could violate the asynchronism postulate, F_k must be a level quantity. On the other hand, any level quantity defined at Ot is of necessity a function of the stable state at this time. This proves the lemma.

The signal superposition theorem can be proved with the help of this lemma.

Theorem 3: On a line in a stable counter carrying pulsed signals, any total signal $f \in \{P\}$ can be resolved into a unique

union of disjoint pulsed quantities in the form

$$f = \bigcup_j (F_j)\ p^{jt}$$

where $F_j \in \{L\}$ are functions of the preceding stable state.

Proof: Consider the signal on the line at any particular event time kt. By the Lemma just proved, the signal is of the form $F_k p^{kt}$, indicating $p^{kt} = 1$. But then, $p^{jt} = 0$ for all $j \neq k$, since $p^{jt} p^{kt} = 0$, (see Theorem 1) and any terms containing p^{jt}, $j \neq k$, contribute nothing to the expression for the signal. Similarly, $p^{kt} = 0$ at any other event time jt, where another unique term $F_j p^{jt}$ will describe the signal. As a result, all terms $F_k p^{kt}$ are disjoint and may be written as a logical union.

To prove uniqueness, assume some expression $\bigcup_j (F'_j) p^{jt}$ which also describes the total signal but F'_j is not necessarily equal to F_j. Then, one can take

$$\bigcup_j (F'_j) p^{jt} = \bigcup_j (F_j) p^{jt} \qquad (2.13)$$

$$\bigcup_j (F_j \oplus F'_j) p^{jt} = 0 \qquad (2.14)$$

Now since the quantities p^{jt} are mutually disjoint, it follows that

$$(F_j \oplus F'_j)\, p^{jt} = 0 \ , \quad \text{for all } j$$

But by definition $p^{jt} = 1$ at each event time jt. Hence $F_j \oplus F'_j$ must vanish at each event time jt, which means that $F_j = F'_j$ for all jt. This, in turn is simply another way of stating that the two different expressions are the same after all, proving uniqueness.

The preceding theorems provide a solid basis for the analysis of an asynchronous counter. In particular, they indicate that given any stable counter, the signal on any pulsed line can for any event time be traced back to its unique source, the transition command p. Furthermore, since this command is applied only once, the expression for the signal for a given time is disjoint from the expression for a different time. Therefore, a general expression for the total signal on a line -- applicable to any event time -- can be given in closed form. Finally, this closed form can be composed entirely of terms referring, (in addition to p, of course) only to the stable state defined at event time Ot.

The mathematical model for subinterval analysis is based on equations 2.2 - 2.7, originally formulated for ordinary differentiation logic, although the expressions for Q' and $\bar{Q}'$ are generally back-dated one subinterval, giving Q and $\bar{Q}$ as functions of past internal states and inputs:

$$Q = S^t + T^t\bar{Q}^t + \bar{R}^t\bar{T}^tQ^t = T^t \oplus \bar{R}^tQ^t \oplus S^t\bar{Q}^t \tag{2.15}$$

$$\bar{Q} = R^t + T^tQ^t + \bar{S}^t\bar{T}^t\bar{Q}^t = T^t \oplus R^tQ^t \oplus \bar{S}^t\bar{Q}^t \tag{2.16}$$

As may be seen, the basic mathematical model is the same as that adopted for synchronous differentiation logic. Indeed, synchronous differentiation logic is simply subinterval logic describing a circuit in which all action takes place in a single significant event. The principal difference between the two logics is that in subinterval logic, quantities such as R^t, Q^t, etc., refer to the values of R, Q, etc., one subinterval ago rather than necessarily at time Ot.

In the basic mathematical model for subinterval analysis, a transition pulse α_Q or β_Q is assumed to be emitted immediately upon application of a pulse causing the transition. However, an inherent delay of t is assumed to be associated with this transition pulse, so that it cannot be used as a logical entity until one subinterval later. A gate, on the other hand, is assumed to pass a pulse with a negligible delay compared to t. Also, the <u>state</u> of a memory unit at an event time is assumed to be the state immediately preceding it, whether a transition takes place or not. Thus, an essentially undelayed pulse Q from a memory unit that is triggered, say,

can be obtained only by AND-gating the trigger pulse with the Q side of the output, while a delayed Q pulse can be obtained simply by differentiating the level output on the Q side (in other words, taking a $_Q$ pulse from the unit). This is demonstrated in Figure 2.4. The logic describing the outputs is

$$u = pQ \tag{2.17}$$

$$v = \beta_Q^t = p^t Q^t \tag{2.18}$$

(Note, however that $vu^t \notin P$; that is, if u were intentionally delayed by t, it would nevertheless not be expected to be synchronized with the inherently delayed pulse v.)

A more involved example for analysis is shown in Figure 2.5. The logic for the signals u, v, and w is:

$$v = \beta_Q^t = u^t Q^t \tag{2.19}$$

$$u = p + w \tag{2.20}$$

$$w = \alpha_R^t = \bar{R}^t v^t \tag{2.21}$$

By simple manipulation among these equations each signal can be expressed in terms of p and its own past values:

$$v = Q^t p^t + Q^t \bar{R}^{2t} v^{2t} \tag{2.19a}$$

$$u = p + \bar{R}^t Q^{2t} u^{2t} \tag{2.20a}$$

$$w = \bar{R}^t Q^{2t} p^{2t} + \bar{R}^t Q^{2t} w^{2t} \tag{2.21a}$$

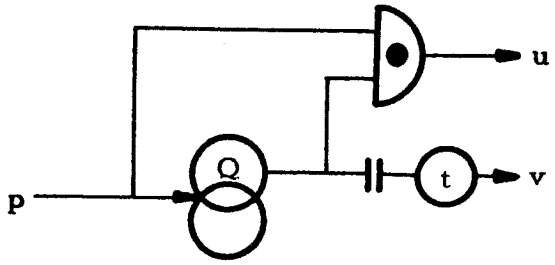


Figure 2.4

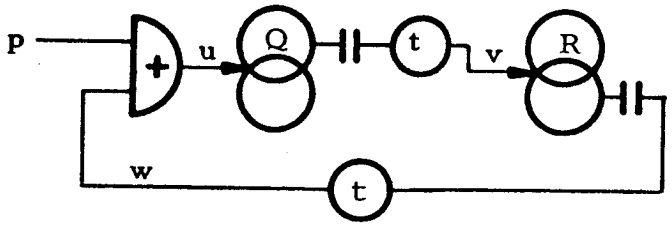


Figure 2.5

If the circuit diagram in Figure 2.5 is now examined, it will be readily seen that these equations can be obtained directly by tracing every pulse signal to its sources, that is, either to p or to its own past around the loop. The procedure is analogous to that of tracing currents or voltages in AC circuit analysis.

Since equations (2.19a-2.21a) are recursive, they may be iterated repeatedly, resulting in an infinite series of terms containing p^{kt}, a description of a seemingly unstable circuit. In actuality, the terms vanish after a certain point as may be found by putting the expressions in the form of equation 2.12, the uniqueness and existence of which is proved in Theorem 3:

$$f = \bigcup_{j} (F_j)\, p^{jt} \qquad (2.12)$$

As will be shown later, equations (2.19a-2.21a) reduce to

$$v = Q^t p^t \qquad (2.19b)$$

$$u = p + \bar{R}^{2t} Q^{2t} p^{2t} \qquad (2.20b)$$

$$w = Q^{2t} \bar{R}^{2t} p^{2t} \qquad (2.21b)$$

indicating the circuit comes to rest in a stable state in a maximum of three events.

To demonstrate further the direct method of subinterval analysis, consider again the 2421 counter shown in Figure 2.2.

For a preliminary setup, each memory transition -- for this

case identified by a trigger signal -- is traced to its sources, with a delay assumed at each flip flop output. The basic setup is:

$$\begin{aligned} T_D &= p \\ T_C &= \beta_D^t + \alpha_A^t \\ T_B &= \beta_C^t + \alpha_A^t \\ T_A &= \beta_B^t \end{aligned} \tag{2.22}$$

In this form, all signals were traced only to their nearest sources. Substituting $\beta_D^t = D^t T_D^t$, $\beta_C^t = C^t T_C^t$, etc., eventually yields the set of recursive equations

$$\begin{aligned} T_D &= p \\ T_C &= p^t D^t + \alpha_A^t \\ T_B &= p^{2t} D^{2t} C^t + \bar{A}^t B^{2t} T_B^{2t} + C^t \bar{A}^{2t} B^{3t} T_B^{3t} \\ T_A &= p^{3t} D^{3t} C^{2t} B^t + B^t C^{2t} \bar{A}^{3t} T_A^{3t} + B^t \bar{A}^{2t} T_A^{2t} \end{aligned} \tag{2.23}$$

This set of equations is of a form particularly suitable for circuit stability analysis, which will be discussed shortly. As may be seen, each transition signal (i.e., trigger signal, for this example) with the exception of T_C, is traced to the command signal p or to its own past through loops. In the case of T_C, no finite equation of this form was possible.

Only a cursory examination suffices to ascertain that, where possible, each equation in (2.23) constitutes the shortest description of every route through the circuit from a signal to its own past or to p. Each product term represents one such route without repetition and without containing any other path represented by a companion term. Such a route may be called a proper path after Hohn, Aufenkamp, and Seshu [15], who used this concept for the determination of non-redundant paths through a weighted directed graph (e.g., a state diagram).

An adaptation of a matrix method used by Hohn, et al will now be developed which will give directly a set of equations such as (2.23). It might be noted that an equivalent method of describing a circuit in the form of (2.23) exists in the regular expression algebra of Kleene [19]. This algebra is capable of handling sequences containing loops by means of the concatenation connective "*" , giving very compact expressions for such sequences (see also Ref. 7). In a recent study, Brzozowski and McCluskey [8] described the use of signal flow graphs to obtain and reduce regular expressions. For purposes of this study, however, the notation and techniques of regular expression algebra are considered less convenient to use than the matrix methods about to be described and will therefore not be employed.

Returning to equation set (2.22), recall that it is permissible to substitute $D^t T_D^t$ for $\overset{t}{D}$, $C^t T_C^t$ for $\overset{t}{C}$, etc., and simply p for T_D. Then, (2.22) can be expressed directly in matrix form:

$$\begin{pmatrix} p \\ T_C \\ T_B \\ T_A \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ D^t & 0 & 0 & \bar{A}^t \\ 0 & C^t & 0 & \bar{A}^t \\ 0 & 0 & B^t & 0 \end{pmatrix} \begin{pmatrix} p^t \\ T^t \\ T_B^t \\ T_A^t \end{pmatrix} \tag{2.24}$$

or

$$\left[T\right] = \left[C\right]^t \left[T\right]^t \tag{2.25}$$

where the multiplication of the matrices is defined by

$$T_i = \bigcup_k C_{ik}^t T_k^t \tag{2.26}$$

The connection matrix $\left[C\right]^t$ is a map of every path of length t between past signals and present signals. If the recursive equation 2.25 is iterated once, the result is

$$T = \left[C\right]^t \left[C\right]^{2t} \left[T\right]^{2t} \tag{2.27}$$

The matrix $\left[C\right]^t \left[C\right]^{2t}$ is, by the same token, a map of every path of length 2t among the signals:

$$\left[C\right]^t\left[C\right]^{2t} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \bar{A}^t B^{2t} & 0 \\ C^t D^{2t} & 0 & \bar{A}^t B^{2t} & C^t \bar{A}^{2t} \\ 0 & B^t C^{2t} & 0 & B^t \bar{A}^{2t} \end{pmatrix} \qquad (2.28)$$

The appearance here of the main-diagonal terms $\bar{A}^t B^{2t}$ and $B^t \bar{A}^{2t}$ indicates the presence of the closed loop between stages A and B. The non-diagonal terms in each row are possible "feeder" signals for the loop, originating outside the loop. The reduction technique in this matrix method consists of the progressive elimination of:

1. Feeder terms containing closed loops of any length
2. Major diagonal terms containing loops specified elsewhere
3. Feeder terms that are expressed in more expanded form elsewhere.

The method also detects the condition where signals cannot be traced to p and their own past values only, in a finite form.

To demonstrate the reduction procedure, consider the matrix $\left[C\right]^t\left[C\right]^{2t} \triangleq \left[\Pi_C\right]^{2t}$.

To begin with, cross out all terms identical to diagonal terms in their own columns. This gets rid of feeder terms containing closed loops, which cannot be reduced further.

For example, in equation (2.28), cross out the term $\bar{A}^t B^{2t}$ in row 2, lying directly above the identical diagonal term of row 3. This is sufficient to indicate that T_C cannot be expressed in terms of p and T_C only, without a detailed stability analysis.

Further reduction can now be undertaken. Since part of a path of length 2t is a path of length t, matrix (2.29) must contain as terminal paths all paths of length t, listed in $\left[C\right]^t$ except those originating one subinterval ago, which are thus not parts of paths of greater length. The simplification can thus proceed by examining $\left[C\right]^t$ and $\left[\Pi_C\right]^{2t}$, in each row of $\left[C\right]^t$ crossing out all terms contained fully in the same row of $\left[\Pi_C\right]^{2t}$. For example, comparing rows 3 of $\left[C\right]^t$ and $\left[\Pi_C\right]^{2t}$ for the 2421 counter,

$$(0 \quad C^t \quad 0 \quad \bar{A}^t) \quad --- \quad (C^t D^{2t} \quad 0 \quad \bar{A}^t B^{2t} \quad C^t \bar{A}^{2t})$$

it can be seen that both C^t and $\bar{A}^t$ are contained fully in the third row of $\left[\Pi_C\right]^{2t}$ and may therefore be crossed out in $\left[C\right]^t$. Comparing rows 2, on the other hand,

$$(D^t \quad 0 \quad 0 \quad \bar{A}^t) \quad --- \quad (0 \quad 0 \quad \cancel{\bar{A}^t B^{2t}} \quad 0)$$

indicates that neither $\bar{A}^t$ nor D^t can be crossed out in row 2 of $\left[C\right]^t$. The reduced matrix $\left[C\right]_r^t$ is simply,

$$\left[C\right]_r^t = \begin{pmatrix} 0 & 0 & 0 & 0 \\ D^t & 0 & 0 & \bar{A}^t \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad (2.29)$$

The reason for encircling $\bar{A}^t$ will be explained shortly.

Next, examine matrix $\left[\Pi_C\right]^{2t}$. Since no loop is to be repeated in the simplification, encircle all terms on the major diagonal and do not use them in further multiplication.

$$\left[\Pi_C\right]^{2t}\left[C\right]^{3t} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ C^tD^{2t} & 0 & \bar{A}^tB^{2t} & C^t\bar{A}^{2t} \\ 0 & B^tC^{2t} & 0 & B^t\bar{A}^{2t} \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 0 \\ D^{3t} & 0 & 0 & \bar{A}^{3t} \\ 0 & C^{3t} & 0 & \bar{A}^{3t} \\ 0 & 0 & B^{3t} & 0 \end{pmatrix}$$

Do not use in further multiplication

$$= \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & C^t\bar{A}^{2t}B^{3t} & 0 \\ B^tC^{2t}D^{3t} & 0 & 0 & B^tC^{2t}\bar{A}^{3t} \end{pmatrix} \qquad (2.30)$$

This matrix can be used to simplify $\left[\Pi_C\right]^{2t}$, in the same way as $\left[\Pi_C\right]^{2t}$ was used to simplify $\left[C\right]^t$. The new reduced matrix is

$$\left[\Pi_C\right]_r^{2t} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ C^tD^{2t} & 0 & \bar{A}^tB^{2t} & 0 \\ 0 & 0 & 0 & B^t\bar{A}^{2t} \end{pmatrix} \qquad (2.31)$$

The nondiagonal terms of this matrix are only terms originating exactly 2 subintervals ago while the diagonal terms describe a cycle of length 2t.

The reduction process continues by iteration so long as nonzero matrices result in the process. At each iteration, the following reduction method is used:

1. In the matrix $\left[\Pi_C\right]^{kt} = \left[\Pi_C\right]^{(k-1)t}\left[C\right]^{kt}$ encircle all diagonal terms. They are not to be used in further iteration multiplications.
2. In $\left[\Pi_C\right]^{kt}$, if any feeder (non-diagonal) term contains a <u>loop term</u> (diagonal term from a given <u>column</u>) -- back-dated or not -- from any matrix $\left[\Pi_C\right]^{jt}$, $1 \leq j \leq k$, cross out that term and <u>encircle the term in $\left[\Pi_C\right]^{(k-1)t}$ which generated it</u>. This term should <u>not</u> have been used in multiplication during the last iteration, so now cross out all other terms in $\left[\Pi_C\right]^{kt}$ that it had generated. (In practice, these are easily located in the same row

as that in which the offending term is located.

3. In $[\Pi_C]^{(k-1)t}$ cross out all terms which are accounted for at greater length in the corresponding rows of $[\Pi_C]^{kt}$.

The matrix that remains is defined as the reduced matrix $[\Pi_C]_r^{(k-1)t}$. Its encircled diagonals identify loops while the other encircled terms identify paths definitely within loops or connected to loops. The terms that are not encircled originate either from p or from larger or more distant loops than the path length can trace.

On the basis of the rules just stated, the analysis of the 2421 counter can be concluded.

Matrix (2.30) is the highest-order significant matrix for this circuit. Any additional iteration produces a zero matrix. The result of the analysis then can be summarized as follows:

$$[T] = [C]_r^t[T]^t + [\Pi_C]_r^{2t}[T]^{2t} + [\Pi_C]_r^{3t}[T]^{3t} + \cdots$$

$$= \bigcup_{k=1}^{N} [\Pi_C]_r^{kt}[T]^{kt} \qquad (2.32)$$

For the 2421 counter, this is identical to the equation set (2.23).

To illustrate the method just described with another example, consider the circuit in Figure 2.6.

The matrix equation for the circuit is

$$\begin{pmatrix} p \\ T_C \\ T_B \\ T_A \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ D^t & 0 & \bar{B}^t & 0 \\ 0 & C^t & 0 & A^t \\ 0 & \bar{C}^t & B^t & 0 \end{pmatrix} \begin{pmatrix} p^t \\ T_C^t \\ T_B^t \\ T_A^t \end{pmatrix} \qquad (2.33)$$

The first iteration gives

$$[\Pi_C]^{2t} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \bar{B}^t C^{2t} & 0 & \bar{B}^t A^{2t} \\ C^t D^{2t} & A^t \bar{C}^t & C^t \bar{B}^{2t} + A^t B^{2t} & 0 \\ \bar{C}^t D^{2t} & B^t C^{2t} & \bar{C}^t \bar{B}^{2t} & B^t A^{2t} \end{pmatrix} \qquad (2.34)$$

The second iteration yields

$$[\Pi_C]^{2t}[C]^{3t} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \cancel{\bar{B}^t A^{2t} C^{3t}} & \cancel{\bar{B}^t A^{2t} B^{3t}} & 0 \\ A^t \bar{C}^{2t} D^{3t} & 0 & A^t \bar{C}^{2t} \bar{B}^{3t} & 0 \\ \cancel{B^t C^{2t} D^{3t}} & \cancel{\bar{C}^t \bar{B}^{2t} C^{3t}} & \cancel{B^t C^{2t} B^{3t}} & \bar{C}^t \bar{B}^{2t} A^{3t} \end{pmatrix} \qquad (2.35)$$

Note the crossed out terms in this matrix. $\bar{B}^t A^{2t} B^{3t}$ in row 2, column 3, contains the loop term $A^t B^{2t}$ for that column, back-dated one subinterval, so it is crossed out and the generating term $\bar{B}^t A^{2t}$ in $[\Pi_C]^{2t}$ is encircled (see matrix 2.37).

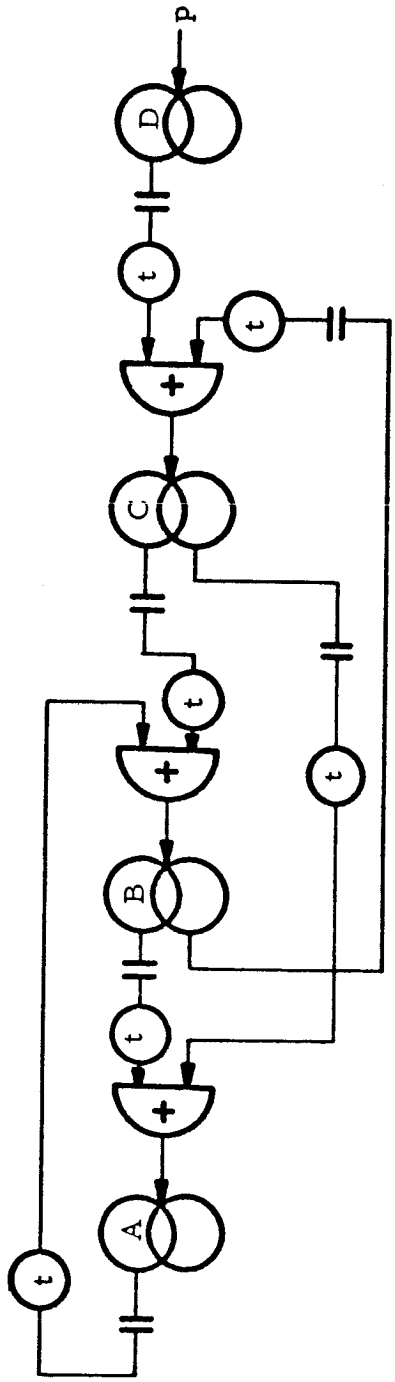


Figure 2.6

Since this term also generated the diagonal term $\bar{B}^t A^{2t} \bar{C}^{3t}$, the latter has been crossed out as well. The same procedure has been applied to the terms of row 4.

A further iteration produces a zero matrix. The reduced matrices for the complete circuit are therefore:

$$\left[C\right]_r^t = \begin{pmatrix} 0 & 0 & 0 & 0 \\ D^t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.36}$$

$$\left[\Pi_C\right]_r^{2t} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \bar{B}^t C^{2t} & 0 & \bar{B}^t A^{2t} \\ C^t D^{2t} & 0 & C^t \bar{B}^{2t} + A^t B^{2t} & 0 \\ \bar{C}^t D^{2t} & B^t C^{2t} & \bar{C}^t \bar{B}^{2t} & B^t A^{2t} \end{pmatrix} \tag{2.37}$$

$$\left[\Pi_C\right]_r^{3t} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ A^t \bar{C}^{2t} D^{3t} & 0 & A^t \bar{C}^{2t} \bar{B}^{3t} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.38}$$

The corresponding equations for the circuit are:

$$T_D = p \text{ (specified at the outset by inspection)} \quad (2.39a)$$

$$T_C = D^t p^t + \bar{B}^t C^{2t} T_C^{2t} + \bar{B}^t A^{2t} T_A^{2t} \quad (2.39b)$$

$$T_B = C^t D^{2t} p^{2t} + A^t \bar{C}^{2t} D^{3t} p^{3t} + (C^t \bar{B}^{2t} + A^t B^{2t}) T_B^{2t} + A^t \bar{C}^{2t} \bar{B}^{3t} T_B^{3t} \quad (2.39c)$$

$$T_A = \bar{C}^t D^{2t} p^{2t} + B^t C^{2t} T_C^{2t} + \bar{C}^t \bar{B}^{2t} T_B^{2t} + B^t A^{2t} T_A^{2t} \quad (2.39d)$$

As may be seen, the method just outlined provides a relatively straight-forward technique for setting up subinterval logic equations, given a circuit diagram or a connection matrix $[C]^t$. The chief advantage is accuracy and the fact that, where possible, the final form is the purely recursive form of equation (2.39c). Where such form is not possible, the fact also becomes readily apparent and the shortest alternate form appears automatically in the result.

The circuits used for examples happened to be composed of T flip flops only. When R S T flip flops are present, the symbol $\Delta_Q = \alpha_Q + \beta_Q$ is substituted for T_Q. The setup method remains the same, but care must be taken, of course, to include in the connection matrix the paths associated with S and R inputs as well as the T inputs. This is discussed at length in Chapter III.

It should also be noted that in both circuits just analyzed,

p was a direct input to one of the stages and the simple substitution $T_D = p$ was made by inspection. Where p is applied through combinational circuitry (which is considered a "path of length zero") such a substitution is no longer possible. Then, the transition vector of the circuit must be expressed in its general form

$$\left[T\right] = \left[T_0\right] p + \left[C\right]^t \left[T\right]^t \qquad (2.40)$$

Here $\left[T_0\right]p$ is the immediate response of the circuit at 0t, namely the response to p by stages connected to p directly or through gating only, while $\left[C\right]^t \left[T\right]^t$ is the response of the circuit to its own transition signals generated one subinterval ago. By iteration, equation (2.40) can be expressed in terms of the circuit's matrix products.

$$T = \bigcup_{k=0}^{N-1} \left[\Pi_C\right]^{kt} \left[T_0\right]^{kt} p^{kt} + \bigcup_{k=1}^{N} \left[\Pi_C\right]_r^{kt} \left[T\right]^{kt} \qquad (2.41)$$

where $\left[\Pi_C\right]^{0t}$ is defined as $\left[I\right]$, the identity matrix.

For example, the action of the circuit in Figure 2.5 would be described by

$$\begin{pmatrix} T_Q \\ T_R \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} p + \begin{pmatrix} 0 & \bar{R}^t \\ Q^t & 0 \end{pmatrix} \begin{pmatrix} T_Q^t \\ T_R^t \end{pmatrix} \qquad (2.42)$$

Expressed in the form of equation (2.41), this is

$$\begin{pmatrix} T_Q \\ T_R \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} p + \begin{pmatrix} 0 \\ Q^t \end{pmatrix} p^t + \begin{pmatrix} \bar{R}^t Q^{2t} & 0 \\ 0 & Q^t \bar{R}^{2t} \end{pmatrix} \begin{pmatrix} T_Q^{2t} \\ T_R^{2t} \end{pmatrix} \tag{2.43}$$

where for this special case $\left[\Pi_C\right]_r^t = \left[0\right]$.

In simultaneous equation form, (2.43) becomes

$$T_Q = u = p + \bar{R}^t Q^{2t} u^{2t} \tag{2.44}$$

$$T_R = v = Q^t p^t + Q^t \bar{R}^{2t} v^{2t} \tag{2.45}$$

which is identical to equation (2.19a) and (2.20a). The quantity $w = \bar{R}^t v^t$ is not necessary for the specification of the system.

Circuit Stability

As was previously mentioned, it may be possible for a circuit to assume a mode of operation from which it is unable to come to rest. If this is the case, the circuit is inherently unstable. Since transition-coupled circuits are particularly susceptible to instability, they must generally be carefully examined to assure stability. Subinterval logic is a useful tool for this purpose.

In the preceding section, it was shown that transitions in the internal state of a circuit can be described in subinterval logic by a finite vector equation of the general form

$$T = \bigcup_{k=0}^{N-1} \left[\Pi_C\right]^{kt} \left[T_Q\right]^{kt} p^{kt} + \bigcup_{k=1}^{N} \left[\Pi_C\right]_r^{kt} \left[T\right]^{kt} \qquad (2.46)$$

This is similar in form to the set of nonlinear difference equations

$$Y_m(nT) = g_m(nT) + \sum_{j=-\infty}^{n-1} \sum_{k=1} f_{mk}(\{Y\}) \; Y_k(jt) \qquad (2.47)$$

where $g_m(nT)$ is an externally applied forcing function analogous to p in (2.46). Indeed, equation (2.32) has many of the properties of (2.47), most significantly a solution composed of a <u>particular solution</u> and a <u>complementary solution</u>. In the case of a counter, the particular solution is the behavior of the circuit in response to the transition command signal p, applied at Ot while the circuit is in a state of rest. The complementary solution is the behavior of the circuit with only transition signals as inputs. Such behavior can be initiated by one or more spurious transition signals, generated when the circuit is first turned on or introduced at any time by stray coupling. If either of these solutions is unstable, the

circuit is considered unstable. Interestingly enough, since the particular forcing function for a counter is a command pulse applied at Ot only, any configuration of signals resulting from it at a later time could equally well have been caused by spurious pulses in the past. Indeed, a spurious pulse could be applied at the input line itself and thus duplicate the action of p. For this reason, the stability of the circuit is established once the stability of the complementary solution is ascertained. This is the reasoning used in the stability tests using subinterval logic.

A circuit or any portion of a circuit can exhibit unstable behavior only if

a. It has one or more unstable feeders, or

b. It contains one or more unstable loops, or

c. Both of the above.

Now the setup technique described in the preceding section provides a clear picture of the loops and feeder signals in a circuit. Thus, to check stability, it is sufficient to "break into" all loops and examine the signals within them -- both feeders and loop signals. If any of these can recur indefinitely in a loop, the loop is unstable. If not, it is stable. In all cases, the stability test seeks instability

rather than stability, because this approach is simpler in Boolean algebra. The ascertainment of stability lies in the failure to establish instability. The procedure is as follows:

To begin with, set up the equations for the circuit, neglecting feeder terms containing p explicitly. If the matrix method is used for the setup, omit columns and rows corresponding to p, if any. For example, the appropriate matrix for the 2421 counter is

$$\left[C\right]^t = \begin{pmatrix} 0 & 0 & \bar{A}^t \\ C^t & 0 & \bar{A}^t \\ 0 & B^t & 0 \end{pmatrix} \qquad (2.48)$$

The justification for disregarding p is that if the circuit is unstable, it will be possible to choose for analysis an event time sufficiently distant from 0t to eliminate all terms of the explicit form $F_k p^{kt}$ and still be able to observe the instability.

Iteration of matrix (2.48) results in loop equations for signals in the circuit. For stability analysis, it is not necessary to examine all the signals in a circuit. Rather, it suffices to examine a set of signals such that every loop in the circuit is "broken into".

A simple way of choosing a set of signals for stability analysis is merely to examine the reduced matrices of the circuit (with rows and columns for p omitted) and find a set of literals which, when replaced by 0, will eliminate all the circled terms. For example, replacing $\bar{A}$ by 0 in the matrices (2.29-2.31) for the 2421 counter eliminates all the circled terms. Hence, if T_A is stable, the entire circuit is stable.

A wider choice of signals for stability analysis can in some cases be found by iteration of the connection matrix without crossing out terms containing loops, described in simplification rule 2 on page 38 and continuing until all distinguishable loops are accounted for. Then, the literals are again sought such that all circled terms are eliminated when these literals are replaced by zero. For example, in the 2421 counter, this procedure yields

$$\text{Modified } \left[\Pi_C\right]_r^t = \left[0\right] \tag{2.49}$$

$$\text{Modified } \left[\Pi_C\right]_r^{2t} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \bar{A}^t B^{2t} & 0 \\ 0 & 0 & B^t \bar{A}^{2t} \end{pmatrix} \tag{2.50}$$

$$\text{Modified } \left[\Pi_C\right]_r^{3t} = \begin{pmatrix} \bar{A}^t B^{2t} C^{3t} & 0 & \bar{A}^t B^{2t} \bar{A}^{3t} \\ 0 & C^t \bar{A}^{2t} B^{3t} & 0 \\ 0 & 0 & B^t C^{2t} \bar{A}^{3t} \end{pmatrix} \qquad (2.51)$$

Examining these matrices, it is clear that not only $\bar{A}$ but also B is a literal whose vanishing eliminates all circled terms. Therefore, if either stages A or B are stable, the entire circuit is stable.

For a second example, consider the reduced matrices (2.36-2.38) on page 42 for the circuit in Figure 2.6. By inspection, it can be seen that setting $\bar{B}$ and B to 0 all circled terms can be eliminated. Hence, the stability of the circuit is determined by the stability of stage B. Note that replacing both $\bar{B}$ and B by 0 is only a notational shortcut in this matrix context to the statement $T_B = 0$. It is not an attempt to circumvent the basic postulates of Boolean Algebra.

Little can be said regarding an "optimum" choice of signals for stability analysis except that as many as possible should be expressible in the purely recursive form (i.e. they should not contain feeder terms from other loops) while the remainder should have all their feeder terms referred to these purely recursive signals. In this way, once the purely

recursive signals are proved stable, all feeder terms arising from them can be disregarded when the remaining signals are analyzed.

For an illustration of stability analysis, the stability of the 2421 counter will be determined by examining the signal T_B. The recursive equation for T_B, from equation (2.23) is written here with the feeder term $p^{2t}D^{2t}C^{t}$ neglected,

$$T_B = \bar{A}^{t}B^{2t}T_B^{2t} + C^{t}\bar{A}^{2t}B^{3t}T_B^{3t} \qquad (2.52)$$

This is now rewritten as

$$\alpha_B + \beta_B = \alpha_A^{t}\beta_B^{2t} + \beta_C^{t}\alpha_A^{2t}\beta_B^{3t} \qquad (2.53)$$

Note, however, that the quantities on the right could not be actually implemented in the form shown since this would violate the asynchronism postulate. They are, rather, a symbolic description of the chains of events that can produce a transition in B and can be used this way only on paper.

Superficially, there appears to be a considerable degree of freedom in equation (2.53) and the existence of a self-perpetuating logic relationship seems almost certain. In actuality, there are many restrictions imposed on α_B and β_B

and the possibilities soon reduce to only a few. The restrictions are:

a. For any stage, α and β can never occur simultaneously:
$$\alpha_Q^{kt}\beta_Q^{kt} = 0 \quad \text{for all } Q \text{ and } k.$$

b. No stage can have two α events or two β events in succession:
$$\alpha_Q^{kt}\alpha_Q^{(k+1)t} = \beta_Q^{kt}\beta_Q^{(k+1)t} = 0 \text{ for all } Q \text{ and } k.$$

c. Every pair of successive β events must be separated by an odd number of α events:
$$\beta_Q^{kt}\beta_Q^{(k+r)t}\left(\overline{\alpha_Q^{(k+1)t} \oplus \cdots\cdots \oplus \alpha_Q^{(k+r-1)t}}\right) = 0$$

d. Every pair of successive α events must be separated by an odd number of β events:
$$\alpha_Q^{kt}\alpha_Q^{(k+r)t}\left(\overline{\beta_Q^{(k+1)t} \oplus \cdots\cdots \oplus \beta_Q^{(k+r-1)t}}\right) = 0$$

Rules b, c, and d, of course, are simply a way of saying that the mutually disjoint α and β events must alternate for a stage, although a pair of events can be separated by any number of "zero events".

The above are only basic restrictions, applicable in all

cases. In addition, other restrictions generally appear for a particular case in the form of requirements for the maintenance of an unstable loop. If these requirements cannot be met, the circuit will be stable.

Returning to equation (2.53), imagine now that an unstable cycle is in progress, having begun an arbitrarily long time ago and destined to continue indefinitely. Picture a "slice" of time that includes the past three event times and all the possible combinations of events that could have taken place during this time. This will determine not only the event about to take place but also the event that follows. The results will, in turn, determine two new consecutive "slices" of time, and so forth. The important question is: Can any combination of events in the present "slice" produce an endless chain of events?

Table 2.1 shows the "slice" preceding an arbitrary event time kt, plus a fairly extensive region of neighboring event times. As may be seen, only four combinations of β's in the slice are permissible from the outset. These are shown as uncircled quantities. The circled quantities are required to be those shown by the basic restrictions previously discussed and each is accompanied by a letter identifying the

Event time	(k-5)t		(k-4)t		(k-3)t		(k-2)t		(k-1)t		kt		(k+1)t		(k+2)t	
Event	β_B	α_B	β_B	α_B	β_B	α_B	β_B	α_B	β_B	α_B	β_B	α_B	β_B	α_B	β_B	α_B
Line 1					0		0		0							
2			(b 0)		1	(a 0)	0		0							
3					0		1	(a 0)	0							
4					0		0		1	(a 0)	(b 0)					
5			(b 0)		1	(a 0)	0	(c 1)	1	(a 0)	(b 0)					

Typical "slice"

Table 2.1

Application of Basic Rules

appropriate restrictions on page 52.

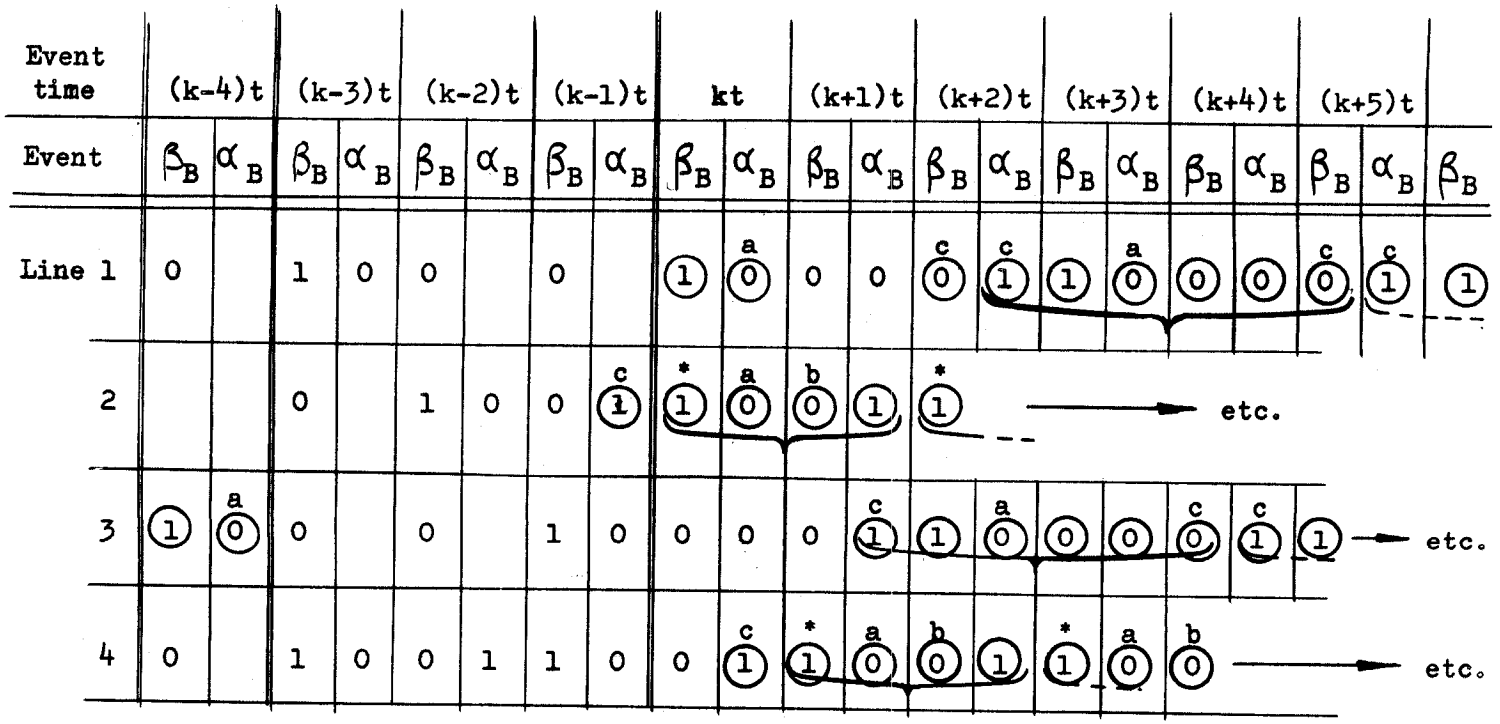
Table 2.2 takes up from Table 2.1 and the new circled quantities result from an application of the equation for $\alpha_B + \beta_B$ to the quantities already shown. In particular, it is an application of the relation that $\alpha_B = \beta_B = 0$ at any event time if β_B was equal to zero both two and three subintervals before. One immediate new result is that <u>three consecutive event times with $\beta = 0$ constitute a stable circuit action</u>. Therefore, the row corresponding to this sequence may be omitted from the list of sequences producing possible instabilities.

Table 2.3 shows the effect of applying the above new restriction to the results already obtained and, of course, reapplying the basic rules where applicable. Notice that lines 1 and 3 now reveal the same fully determined cycle, one line differing from the other in phase only. This means that the event sequence $(\beta - \bar{\alpha}) \rightarrow (\bar{\beta} - \emptyset) \rightarrow \bar{\beta}$ always identifies the cycle of lines 1 and 3. Now notice that both lines 2 and 4 end with $(\beta - \bar{\alpha}) \rightarrow (\bar{\beta} - \emptyset)$ in the table. If the next event in either line is $\bar{\beta}$, then the result will be the same sequence as in lines 1 and 3. Therefore, for a different cycle, the next event must be β, which changes $\emptyset$

Event time	(k-5)t		(k-4)t		(k-3)t		(k-2)t		(k-1)t		kt		(k+1)t		(k+2)t	
Event	β_B	α_B	β_B	α_B	β_B	α_B	β_B	α_B	β_B	α_B	β_B	α_B	β_B	α_B	β_B	α_B
Line 1					0		0		0		⓪	⓪	⓪	⓪	⓪	⓪ →
2			0		1	0	0		0				⓪	⓪		
3					0		1	0	0							
4					0		0		1	0	⓪	⓪	ⓒ⓪			
5			0		1	0	0	1	1	0	0					

Table 2.2

Preliminary Application of Recursive Relation



* - Required to avoid entering cycle of lines 1 and 3

Table 2.3

Application of Requirements for Possible Instability

Brackets indicate possible cycles

to α because of basic rule c. Finally, in order to keep from entering the cycle of lines 1 and 3, the basic cycle $(\beta - \bar{\alpha}) \rightarrow (\bar{\beta} - \alpha)$ must keep repeating indefinitely.

To sum up the results of this study, only two distinguishable cycles could possibly be exhibited in the case of instability and they are $(\beta - \bar{\alpha}) \rightarrow (\bar{\beta} - \bar{\alpha}) \rightarrow (\bar{\beta} - \bar{\alpha})$ and $(\beta - \bar{\alpha}) \rightarrow (\bar{\beta} - \alpha)$.

At this point, it is not difficult to determine that neither of these cycles can perpetuate itself. The first cycle can be described by the equations

$$\begin{aligned} \beta_B &= \beta_B^{3t} = \alpha_B^t \\ \alpha_B &= \alpha_B^{3t} = \beta_B^{2t} \end{aligned} \tag{2.54}$$

Now choose some point where $\beta_B = 1$ for reference. Then

$$\beta_B \beta_B^{3t} \beta_B^{6t} = 1 \tag{2.55}$$

Applying the recursion formula (2.53) to this equation gives

$$\beta_C^t \alpha_A^{2t} \beta_C^{4t} \alpha_A^{5t} \beta_B^{6t} = 1 \tag{2.56}$$

Then,

$$\alpha_A^{2t} = \alpha_A^{5t} = 1 \qquad (2.57)$$

By basic rule d, this, in turn, implies that $\beta_A^{3t} + \beta_A^{4t} = 1$ and the recurrence relationship for stage A makes this equivalent to saying that $\beta_B^{4t} + \beta_B^{5t} = 1$. But since $\beta_B^{3t} = \beta_B^{6t} = 1$, this would require at least two consecutive β events for stage B, a clear violation of basic rule b on page 52. As a result, the cycle described by equations (2.54) is impossible.

The second cycle indicated on page 58 is described by

$$\begin{aligned} \beta_B &= \beta_B^{2t} \\ \alpha_B &= \alpha_B^{2t} \end{aligned} \qquad (2.58)$$

Again choosing a reference point where $\beta_B = 1$ permits the statement

$$\beta_B \beta_B^{2t} \beta_B^{4t} = 1 \qquad (2.59)$$

or

$$\alpha_A^t \alpha_A^{3t} \beta_B^{4t} = 1 \qquad (2.60)$$

By basic rule d, on page 52, this means that $\beta_A^{2t} = 1$, and by the recurrence relation for stage A, this indicates $\beta_B^{3t} = 1$. But this is impossible, since $\beta_B^{2t} = \beta_B^{4t} = 1$, requiring that $\alpha_B^{3t} = 1$. Therefore, this second cycle is also not possible.

The conclusion of this brief analysis, then is that the 2421 counter is stable in response to any finite sequence of signals, that is, either in the process of normal count or in response to any transient disturbance.

In the preceding study, three tables were used only to demonstrate the development of a cycle table in its different stages of progress. In practice, a single table might have sufficed and identification of the rules applicable to circled figures would have been omitted. Note in particular the economy of this method of analysis: At first, only basic rules were applied. When their usefulness alone was exhausted a simple part of the recurrence relationship was applied and finally, an observed result of this application was used to complete the development of the table. In this way, the more involved logical relationships entailed in the recursion formulas for the circuit were avoided until they were actually needed, by which time their application was limited to the investigation of only two possible, well defined cycles -- both of which turned out to be incapable of recurring indefinitely.

The simplicity of the proceding analysis served to demonstrate the basic attack on the general stability problem. It should not, on the other hand, be mistakenly assumed to be

shared by all counter-type structures; the sad fact is that a thorough stability analysis is in general a tedious task. Whereas the response to a command pulse p, given an initial state of rest for the circuit, involves the relatively simple problem of following a single signal through the circuit, the complete circuit action will usually be based on a large set of possible initial conditions, any of which may appear in the circuit as a transient disturbance. A stability analysis is therefore a necessary as well as laborious task. However, the analysis method just described makes this task as simple as possible, by using at any step only as much of the natural and imposed constraints as is needed.

To show an example of a moderately complicated stability analysis, reconsider the circuit in Figure 2.6. As was previously indicated, it is sufficient to examine the transitions of stage B to determine the stability of the entire circuit. Neglecting terms originating from the command pulse p gives

$$\alpha_B + \beta_B = \beta_C^t \alpha_B^{2t} + \beta_A^t \beta_B^{2t} + \beta_A^t \alpha_C^{2t} \alpha_B^{3t} \tag{2.61}$$

As in the previous example, a cycle chart is constructed containing a "slice of time" three subintervals long and the various possible α_B and β_B signals present in that slice. This

appears in Table 2.4. Since the transitions of stage B depend this time on past values of α_B as well as β_B, there is a greater number of possibilities for distinguishable results.

The chart is now filled in as follows:

1. Apply all basic rules, where applicable.
2. Notice from recursive equation (2.61) that the $\bar{\alpha}^{3t} \longrightarrow (\beta^{2t} - \bar{\alpha}^{2t})$ sequence produces no transition at the reference time. Up-date and back-date this feature throughout the chart. Notice that this eliminates line 2 (contradiction of recursion equation) and lines 1 and 12 (response dies out).
3. A conclusion from the data thus far is that an $\bar{\alpha} \longrightarrow (\bar{\beta} - \bar{\alpha}) \longrightarrow (\bar{\beta} - \bar{\alpha})$ sequence stops all circuit action. Since the number of consecutive zeros in the chart must be even everywhere (because of basic rules c and d) this actually implies a sequence of six zeros in the table. Hence, the maximum number of consecutive zeros at any point must be 4, if circuit action is to be kept going. Enter this feature into the chart, putting encircled 1's and 0's where necessary, to keep the number of consecutive zeros even and equal to 4 or less.

Event time		(k-3)t		(k-2)t		(k-1)t		kt		(k+1)t		(k+2)t		(k+3)t		(k+4)t	
Event	α	β	α	β	α	β	α	β	α	β	α	β	α	β	α	β	α
Line 1		~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	signal dies out					
2	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~1~~	contradiction									
3	1	0	0	0	0	1	0	0	0	⓪	①	0	0	0			
4	0	0	0	0	1	0	0	0									
5	0	0	0	0	1	1	0	0									
6		0	0	1	0	0	0	⓪	①	0	0	0					
7	1	0	0	1	0	0	1	0									
8	0	0	1	0	0	0	0	①	⓪	0	0	⓪	①	⓪	⓪	0	
9	0	0	1	0	0	1	0	0									
10	0	0	1	1	0	0	0	⓪	①	0	0	0					
11	0	0	1	1	0	0	1	0									
12		~~1~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	signal dies out					
13		1	0	0	0	0	1	0									
14		1	0	0	1	0	0										
15		1	0	0	1	1	0	0									

Table 2.4

The result of steps 1-3 is Table 2.4.

4. In Table 2.4 notice that 6 of the 12 sequences end with the sequence

$$(\bar{\beta} - \bar{\alpha}) \rightarrow (\bar{\beta} - \alpha) \rightarrow (\bar{\beta} - \bar{\alpha}) \rightarrow (\phi - \bar{\alpha}).$$

Now if this sequence can be shown to be stable, a good portion of the problem will be solved. To do this refer to equation 2.61 in greater detail and enter the possible instabilities in a new table. This is shown in Table 2.5. As may be seen, two sequences can be distinguished, depending on the response of stage C to the α_B^{2t} pulse. In keeping with the spirit of this analysis technique, the use of more complicated logic relationships will be avoided by simply producing the two possible sequences forcibly and observing the results.

One possibility consists of setting stage B (producing an α_B signal) while stages C and A are in the 1 state. This produces the following action:

$$(101) \rightarrow (111) \rightarrow (110) \rightarrow (100) \rightarrow (000) \rightarrow (010) \rightarrow (011) \rightarrow (111)$$

The last state is stable. The other possibility is artificially produced by setting stage B while stage C is in the 0 state and stage A is in the 1 state; the result is

Event time	(k-3)t		(k-2)t		(k-1)t		kt		(k+1)t		(k+2)t		(k+3)t	
Event, referred to kt	β^{3t}	α^{3t}	β^{2t}	α^{2t}	β^{t}	α^{t}	β	α	β'	α'	β''	α''	β'''	α'''
Occurrence of event	0	0	0	1	0	0	β_C^t	0	$\beta_A\alpha_C^t$	0	0	$\beta_A'\beta_C^t$	0	$\beta_A''\beta_A\alpha_C^t$ →
Possible sequence #1	0	0	0	1	0	0	1	0	0	0	0	(1)	0	0 →
Possible sequence #2	0	0	0	1	0	0	0	0	(1)	0	0	0	0	1 →

Table 2.5

$$(100)\rightarrow(110)\rightarrow(111)\rightarrow(011)\rightarrow(011)\rightarrow(101)$$

the last state being stable. Thus, the sequence $(\bar{\beta} - \bar{\alpha}) - (\bar{\beta} - \alpha) - (\bar{\beta} - \bar{\alpha}) - (\phi - \bar{\alpha})$ ..... leads to a stable response and all rows in Table 2.4 ending with it can be disregarded.

Next, consider the remaining rows. Write each twice in a new table, entering a 1 and a 0, respectively, as the first entry that is thus far uncertain. This is shown in Table 2.6. Some of the cases will be seen to lead to the sequence already identified as stable and may therefore be crossed out. Reapply these new results to the rows still in doubt. The startling result is that only two new sequences emerge which are independent of each other and at the same time do not lead to the one already shown to be stable. They are

$$(\bar{\beta} - \alpha)\rightarrow(\beta - \bar{\alpha})\rightarrow(\bar{\beta} - \alpha)\rightarrow(\phi - \bar{\alpha}) \text{ and}$$

$$(\bar{\beta} - \bar{\alpha})\rightarrow(\beta - \bar{\alpha})\rightarrow(\bar{\beta} - \alpha)\rightarrow(\phi - \bar{\alpha}).$$

These can now be investigated in the same way as was done in Table 2.5. The results are that the first sequence is artificially produced by triggering stages B and C while the circuit is in the 101ϕ state or triggering A, B and C while the

Event time	(k-3)t		(k-2)t		(k-1)t		kt		(k+1)t		(k+2)t		(k+3)t		(k+4)t	
Event	β	α	β	α	β	α	β	α	β	α	β	α	β	α	β	α
Line 5	~~0~~	~~0~~	~~0~~	~~1~~	~~1~~	~~0~~	~~0~~	~~0~~	~~0~~	~~1~~	~~0~~	~~0~~		0		
	0	0	0	1	1	0	0	1		0	← new sequence a					
Line 7	0	0	1	0	0	1	0	0	1	0	0	1		0		
	0	0	1	0	0	1	1	0	0	1		0				
													new sequence b			
Line 9	~~0~~	~~1~~	~~0~~	~~0~~	~~1~~	~~0~~	~~0~~	~~0~~	~~0~~	~~1~~	~~0~~	~~0~~		0		
	0	1	0	0	1	0	0	1		0						
Line 11	0	1	1	0	0	1	0	0	1	0	0	1		0		
	0	1	1	0	0	1	1	0	0	1		0				
Line 14	~~1~~	~~0~~	~~0~~	~~1~~	~~0~~	~~0~~	~~0~~	~~0~~	~~1~~	~~0~~	0	0	0	1	0	0
	1	0	0	1	0	0	1	0	0	1		0				
Line 15	~~1~~	~~0~~	~~0~~	~~1~~	~~1~~	~~0~~	~~0~~	~~0~~	~~0~~	~~1~~	~~0~~	~~0~~		0		
	1	0	0	1	1	0	0	1		0						

Table 2.6

Elimination of Sequence Analyzed in Table 2.5

(Line numbers are those of Table 2.4)

circuit is in the 100Ø state. The response is stable in both cases. The second sequence is generated by triggering stages A and C while the circuit is in the 110Ø state or triggering A and D while the circuit state is 1111. (This excitation can serve only as a starting input, since stage D is not part of any loop). Here, too, the response is stable, establishing the stability of the circuit.

As in the case of the 2421 counter, the principal technique in the stability analysis just performed has been the progressive application of basic and imposed constraints to the cycle tables until only three distinct sequences of transitions on the part of a stage were identified as possible manifestations of unstable behavior. Associated with these sequences were only six combinations of starting states and inputs which could produce them artifically, and the behavior of the circuit was easily checked for all six combinations. Contrast the simplicity of this methodical technique with that of trying out every one of the 240 possible sets of starting states and inputs (16 possible starting states X 15 possible nontrivial trigger inputs).

Once the stability of a circuit has been established, it is possible to specify in detail the behavior of the circuit

in response to the externally applied command signal and thus answer the important question "What does the circuit do?" The next chapter will be devoted to answering this question - regardless of when it is asked.

III. SUBINTERVAL ANALYSIS OF THE STABLE CIRCUIT

As was seen in the last chapter, subinterval logic is capable of rendering a clear and compact description of the various events that comprise the circuit's transition from one stable state to the next. If the circuit's stability is in question, recursion equation sets such as (2.23) (for the 2421 counter) and (2.39) (for the counter in Figure 2.6) furnish the most useful description of the circuit's behavior. If, on the other hand, the circuit is known to be inherently stable, a far more informative and directly usable description is possible. The development of such a description will now be considered in some detail.

Suppose a circuit is examined by the methods discussed in Chapter II and a set of transition equations is determined, having the general form

$$[T] = \bigcup_{k=0}^{N-1} [\Pi_C]^{kt}[T_Q]^{kt}\, p^{kt} + \bigcup_{k=1}^{N} [\Pi_C]_r^{kt}[T]^{kt} \qquad (3.1)$$

Suppose further that a stability analysis is performed and the circuit is found to be stable. Then, a state of rest may be assumed at event time Ot, when a command signal p is applied and likewise, the transition may be assumed to consist

of a finite number of events. By Theorem 3 (see page 25) the equation set represented in vector form in (3.1) can then be written in the form

$$T = \bigcup_{k=0}^{M} [T_k] p^{kt} \qquad (3.2)$$

where $[T_k]$ are now vectors associated with the state of the circuit at Ot, (kt subintervals ago), and Mt designates the last significant event time.

The form of equations (3.2) shows graphically how simple it is to single out any event time (jt) and specify the significant events at this time.

The intersection of $p^{jt}[I]$ and $[T]$,

$$p^{jt}[I][T] = p^{jt}[T] = p^{jt} \bigcup_{k=1}^{M} [T_k] p^{kt} \qquad (3.3)$$

can be seen to reduce simply to

$$p^{jt}[T] = [T_j] p^{jt} \qquad (3.4)$$

since $p^{jt}p^{kt} = 0$ for $j \neq k$.

But this is simply the set of events occurring at event time jt. Thus, the circuit's action at any event time is

simply the projection of the complete transition vector onto the p^{jt} function and the behavior of the circuit can be easily analyzed by beginning with the p "component" and proceeding to p^{t}, p^{2t}.....etc., "components". When all the components have been found -- and they can be, since the number of events in a stable circuit is finite -- the analysis is complete. Not only does the result provide a clear event-by-event picture of any transition the circuit makes but it also determines the stable state eventually reached by the circuit, which is often not known. This is found by the relation

$$\left[Q^{k}\right] = \left[Q^{k-1}\right] \oplus \sum_{j=0}^{M} \left[T_j\left(\left[Q^{k-1}\right]\right)\right] \quad (3.5)$$

In the above equation, $\left[Q^{k}\right]$ and $\left[Q^{k-1}\right]$ are the kth and (k-1)th stable internal state vectors, respectively, $\sum$ is the multiple ring sum operation and $\left[T_j\left(\left[Q^{k-1}\right]\right)\right]$ are the transition vectors $\left[T_j\right]$ evaluated for $\left[Q^{k-1}\right]$. In other words, for the analysis method to be useful, the general expressions for $\left[T_j\right]$ must be referred to the state of the circuit at Ot, which is assumed to be known. For example, a typical transition vector might be

$$\left[\bar{T}_4\right]p^{4t} = \begin{pmatrix} 0 \\ \bar{A}^{4t}{}_B{}^{4t}{}_C{}^{4t}{}_D{}^{4t} \\ A^{4t}{}_B{}^{4t}{}_{\bar{C}}{}^{4t}{}_D{}^{4t} \\ A^{4t}{}_B{}^{4t}{}_C{}^{4t}{}_{\bar{D}}{}^{4t} \end{pmatrix} p^{4t} \qquad (3.6$$

In contrast, the setup techniques presented thus far do not yield directly expressions of this form when expanded in terms of their p^{jt} components but are rather composed of terms of the type $A^{t}{}_B{}^{2t}{}_{\bar{C}}{}^{3t}{}_D{}^{4t}p^{4t}$. Direct use of such terms entails the obvious difficulty that $A^{t}p^{4t}$ is A at event time 3t, $B^{2t}p^{4t}$ is B at event time 2t, and so on. These quantities, on the other hand, are not known explicitly, and converting them computationally into equivalent expressions referred to the state of the circuit at 0t may be a cumbersome and discouraging task involving tracing back all previous action of the circuit. What is needed, of course, is a simple technique that progressively performs this conversion, eliminating unknown terms as they occur. Such a method will now be presented. Since the reasoning underlying the method is lengthy, if simple, it will be discussed only after the technique has been fully presented and illustrated.

The first thing that is needed is a convenient shorthand notation for certain quantities. Let any level quantity referred to event time Ot at time kt be simply underlined rather than shown with the superscript kt. For example, $A^{3t}p^{3t} \rightarrow \underline{A}p^{3t}$. Further, let any level quantity usually shown with superscript t simply be shown without any superscript or underline. For example $A^{t}p^{3t} \rightarrow Ap^{3t}$. The analysis method now proceeds as follows (See layout illustration in Table 3.1):

1. Write down the connection matrix $\left[C\right]^t$, leaving off the superscripts t, and complementing all literals originally not superscripted (e.g., terms corresponding to S or R inputs). For example, AB^t becomes $\bar{A}B$. (This will be explained shortly).
2. Immediately to the right of $\left[C\right]^t$, write down the column vector $\left[T_0\right]$, which is called the <u>zeroth transition vector</u>. This is simple to find, since it is the response of the circuit <u>at Ot</u>, and thus involves nothing more than locating all memory inputs connected to p, either directly or through gating. For example, suppose p is connected to a four-stage circuit in such a way as always to trigger

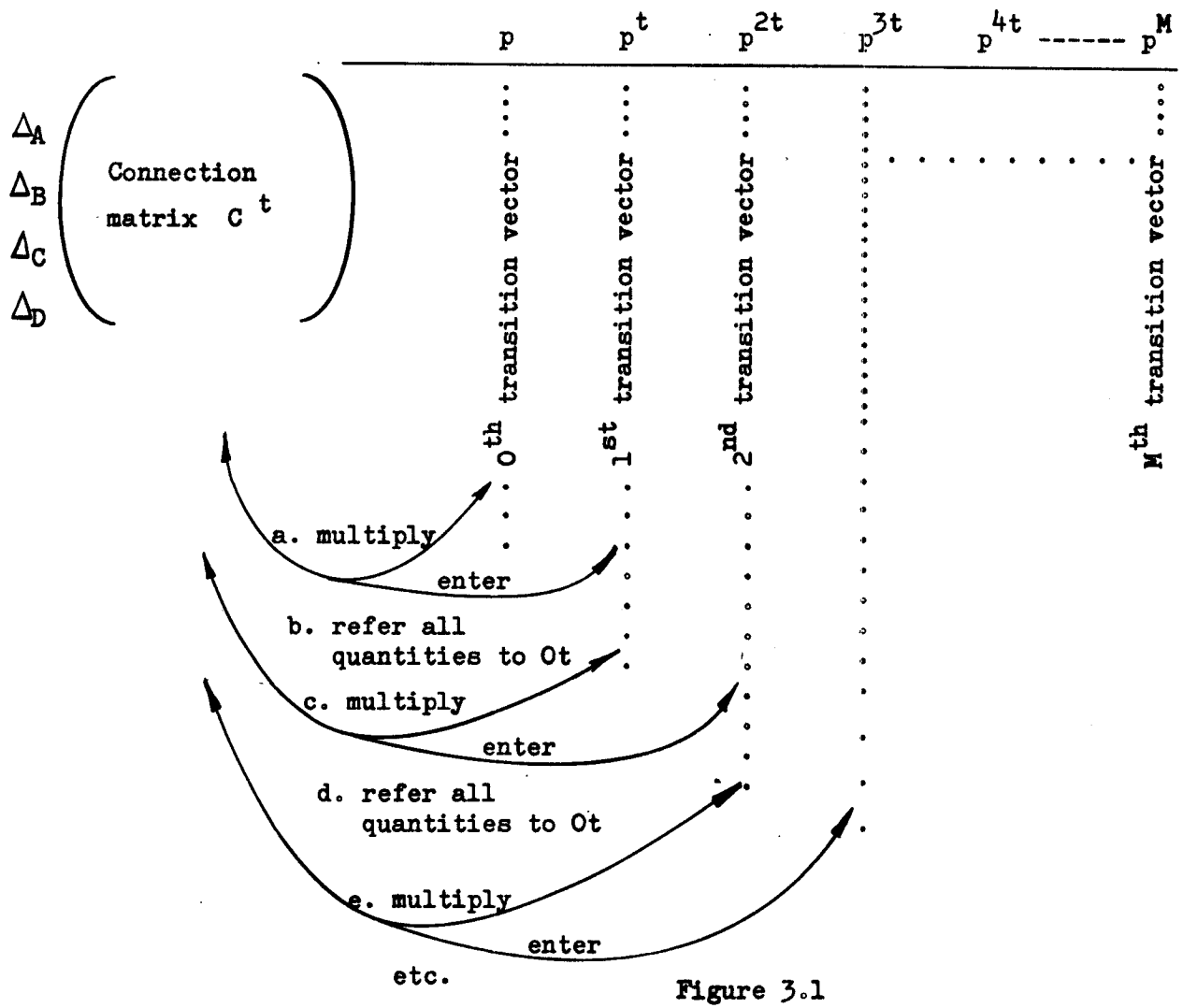

p
p^t
p^2t
p^3t
p^4t
------ p^M
Δ_A
Δ_B
Δ_C
Δ_D
Connection matrix C^t
0th transition vector ....
1st transition vector ....
2nd transition vector ....
Mth transition vector ....
a. multiply
enter
b. refer all quantities to Ot
c. multiply
enter
d. refer all quantities to Ot
e. multiply
enter
etc.

Figure 3.1

D, and also set A if B = 1. Then the zeroth transition vector T_0 is defined by:

$$[T_0]\,p = \begin{pmatrix} \Delta_D \\ \Delta_C \\ \Delta_B \\ \Delta_A \end{pmatrix} p = \begin{pmatrix} \underline{1} \\ 0 \\ 0 \\ \underline{B\bar{A}} \end{pmatrix} p \qquad (3.7)$$

Note that the nonzero quantities inside $[T_0]$ are underlined since they refer to conditions at Ot.

3. Above the zeroth column vector, write "p" identifying the event time. Next to p, write "p^t ", "p^{2t} ", "p^{3t} ", etc., thus labeling columns to be filled in by succeeding transition vectors.

4. Under "p^t " enter the first transition vector obtained by

$$[T_1] = [C]^t\,[T_0] \qquad (3.8)$$

Now notice that the literals originating from $[C]^t$ in each product term of T_1 are not underlined. These must be converted to their underlined equivalents. Take one literal at a time and perform the following simple operation:

$$[T_k] = [C]^t [T_{k-1}] \qquad (3.10)$$

In $[T_k]$, take one literal Q of a product term at a time and note its underlined portion ($\underline{A}$). Consult the Δ_Q items in all previous transition vectors, $[T_0]$ through $[T_{k-1}]$, (these form a row in the gradually growing array of column vectors). If the number of items implying ($\underline{A}$) is even, complement Q and underline it; if odd, simply underline it. Cancel resulting terms of the type $\underline{AB\bar{C}\bar{B}}$ as they occur.

8. Continue step 7 until a zero vector results. The analysis is then completed.

The result yields directly all the terms for the expansion

$$[T] = \bigcup_{k=0}^{M} [T_k] p^{kt} \qquad (3.11)$$

properly referred to conditions at Ot, so that given a starting state, it is simple to determine whether or not a quantity Q undergoes a transition at some event time kt by simply examining the Δ_Q item in $[T_k]$. If this item does not imply the

$$[T_k] = [C]^t [T_{k-1}] \qquad (3.10)$$

In $[T_k]$, take one literal Q of a product term at a time and note its underlined portion ($\underline{A}$). Consult the Δ_Q items in all previous transition vectors, $[T_0]$ through $[T_{k-1}]$, (these form a row in the gradually growing array of column vectors). If the number of items implying ($\underline{A}$) is even, complement Q and underline it; if odd, simply underline it. Cancel resulting terms of the type $\underline{ABC\bar{B}}$ as they occur.

8. Continue step 7 until a zero vector results. The analysis is then completed.

The result yields directly all the terms for the expansion

$$[T] = \bigcup_{k=0}^{M} [T_k] \, p^{kt} \qquad (3.11)$$

properly referred to conditions at Ot, so that given a starting state, it is simple to determine whether or not a quantity Q undergoes a transition at some event time kt by simply examining the Δ_Q item in $[T_k]$. If this item does not imply the

starting state, no transition occurs; if it implies the starting state, a transition does occur.

At any time, including at the end of the transition action, the state of the circuit can be determined by counting transitions corresponding to the starting state. If their number is odd for a literal, the literal is complemented. If even, it is the same as at 0t.

At this point it will be best to illustrate the technique just described with a concrete example. This will be the subinterval analysis of the circuit in Figure 2.6, whose stability was established at the conclusion of Chapter II.

Part of the analysis is performed in Table 3.2. Comments are added as needed. The entire analysis is repeated in more compact form in Table 3.3.

Now as to the interpretation of the results in Table 3.3. The quantities appearing in the table to the right of the connection matrix are the coefficients of the p^{kt} terms shown above them. For example, row 3 is interpreted as:

$$\Delta_B = (\underline{CD})p^{2t} + (\underline{A\bar{C}D})\, p^{3t} + (\underline{ABCD})p^{4t} + (\underline{A\bar{B}D})p^{5t} \tag{3.12}$$

which is equivalent to

$$\Delta_B = (CD)^{2t}p^{2t} + (A\bar{C}D)^{3t}p^{3t} + (ABCD)^{4t}p^{4t} + (A\bar{B}D)^{5t}p^{5t} \tag{3.13}$$

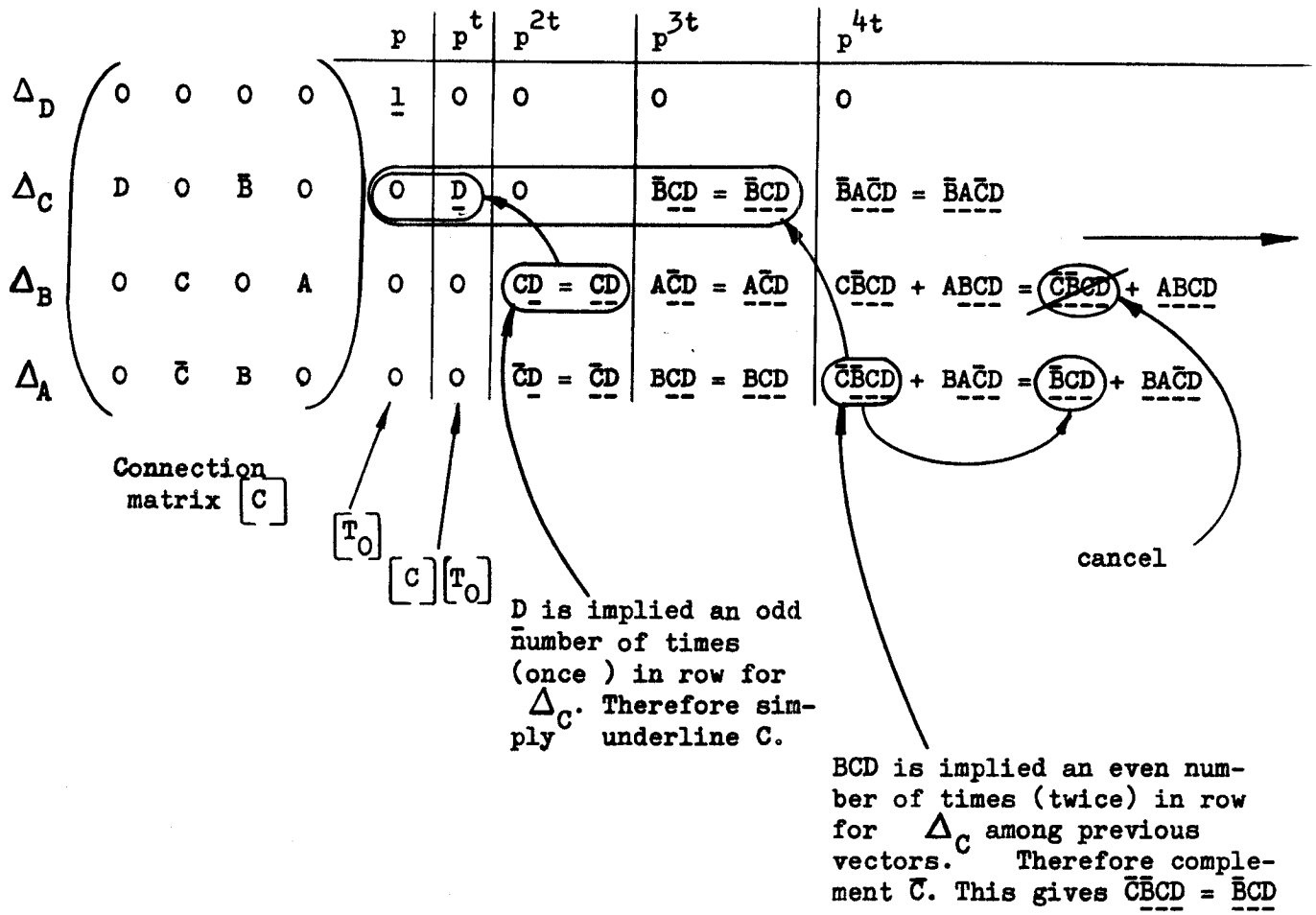

p
p^t
p^2t
p^3t
p^4t
Δ_D
Δ_C
Δ_B
Δ_A
0 0 0 0
D 0 B̄ 0
0 C 0 A
0 C̄ B 0
1 0 0 0 0
0 D 0 B̄CD = B̄CD B̄AC̄D = B̄AC̄D
0 0 CD = CD AC̄D = AC̄D CB̄CD + ABCD = C̄B̄CD + ABCD
0 0 C̄D = C̄D BCD = BCD C̄B̄CD + BAC̄D = B̄CD + BAC̄D
Connection matrix [C]
[T_0]
[C][T_0]
D is implied an odd number of times (once) in row for Δ_C. Therefore simply underline C.
BCD is implied an even number of times (twice) in row for Δ_C among previous vectors. Therefore complement C̄. This gives C̄B̄CD = B̄CD
cancel

Table 3.2

					p	p^t	p^{2t}	p^{3t}	p^{4t}	p^{5t}	p^{6t}	
Δ_D	0	0	0	0	$\underline{1}$	0	0	0	0	0	0	0
Δ_C	D	0	$\bar{B}$	0	0	$\underline{D}$	0	$\underline{\bar{B}CD}$	$\underline{\bar{B}A\bar{C}D}$	$\underline{ABCD}$	0	0
Δ_B	0	C	0	A	0	0	$\underline{CD}$	$\underline{A\bar{C}D}$	$\underline{ABCD}$	$\underline{A\bar{B}D}$	0	0
Δ_A	0	$\bar{C}$	B	0	0	0	$\underline{\bar{C}D}$	$\underline{BCD}$	$\underline{\bar{B}CD}+\underline{BA\bar{C}D}$	0	$\underline{BACD}+\underline{A\bar{B}D}$	0

Table 3.3

Much information can be gained from Table 3.3. For example, it is clear that the longest response for the circuit takes 7 event times and corresponds to initial states ABCD, $A\bar{B}CD$, or $A\bar{B}\bar{C}D$. Stage A is the laggard in these transitions.

The shortest transition takes only one event and that is an α_D for any starting state having a $\bar{D}$.

Stage C can never undergo a transition at event time 2t.

Stage A can never undergo a transition at event time 5t.

For an initial state with $\bar{A}$, stage B reaches its new stable state $\underline{B} \oplus \underline{CD}$ relatively early (after a single transition at event time 2t). Otherwise, it may undergo its last transition as late as event time 5t.

The preceding are only examples of circuit characteristics that may be read off directly from a table such as Table 3.3, and are intended to give insight into the wealth of information contained in such a table -- despite its compactness.

Perhaps the most useful feature of Table 3.3 is the ease with which it can be used to find successive stable states. What is used is simply the relation

$$\left[Q^k\right] = \left[Q^{k-1}\right] \oplus \sum_{j=0}^{M} \left[T_j\left(\left[Q^{k-1}\right]\right)\right] \qquad (3.14)$$

as previously indicated. What this means in practice is that given an initial circuit state, one counts for each literal the number of transitions in the row corresponding to that literal. This is, of course, the number of terms implying the given initial state. If the number of transitions is odd the net effect is complementation. If even, the net effect is that of no transition.

For example, the successive states for the counter in Figure 2.6 are easily calculated in this way from Table 3.3.

Initial Stable State	Next Stable State
A B C D	A B C D
0 0 0 0	0 0 0 1
0 0 0 1	1 0 1 0
0 0 1 0	0 0 1 1
0 0 1 1	1 1 1 0
0 1 0 0	0 1 0 1
0 1 0 1	1 1 1 0
0 1 1 0	0 1 1 1
0 1 1 1	1 0 0 0
1 0 0 0	1 0 0 1
1 0 0 1	1 0 0 0
1 0 1 0	1 0 1 1
1 0 1 1	1 0 1 0
1 1 0 0	1 1 0 1
1 1 0 1	1 0 1 0
1 1 1 0	1 1 1 1
1 1 1 1	1 1 1 0

The state diagram for the circuit is shown in Figure 3.1. As may be inferred from this odd diagram, the practical usefulness of this circuit is doubtful. However, the circuit did serve well to demonstrate the use of the analysis technique just discussed.

In passing, it might be interesting to apply the subinterval analysis method to the extremely simple circuit shown in Figure 2.5 on page 30. (Its stability is easily shown and will not be worked out.) The connection matrix and transition vectors are

$$\begin{array}{c|cc|cccc} & & & p & p^{t} & p^{2t} & p^{3t} \\ \hline \Delta_Q & 0 & \bar{R} & \underline{1} & 0 & \bar{R}\underline{Q} & 0 \\ \Delta_R & Q & 0 & 0 & \underline{Q} & 0 & Q\bar{R}\underline{Q} = \cancel{\bar{Q}\bar{R}\underline{Q}} \end{array}$$

This indicates

$$\begin{aligned} \Delta_Q &= p + \bar{R}^{2t} Q^{2t} p^{2t} \\ \Delta_R &= Q^{t} p^{t} \end{aligned} \qquad (3.15)$$

which was asserted in Equations (2.19b) and (2.20b).

Thus far in this chapter, subinterval analysis has been presented in "recipe" fashion, simply because the bookkeeping scheme illustrated has been found from experience to be the most practical among a very wide variety of possible ones. Detailed explanation of every step in the method would have

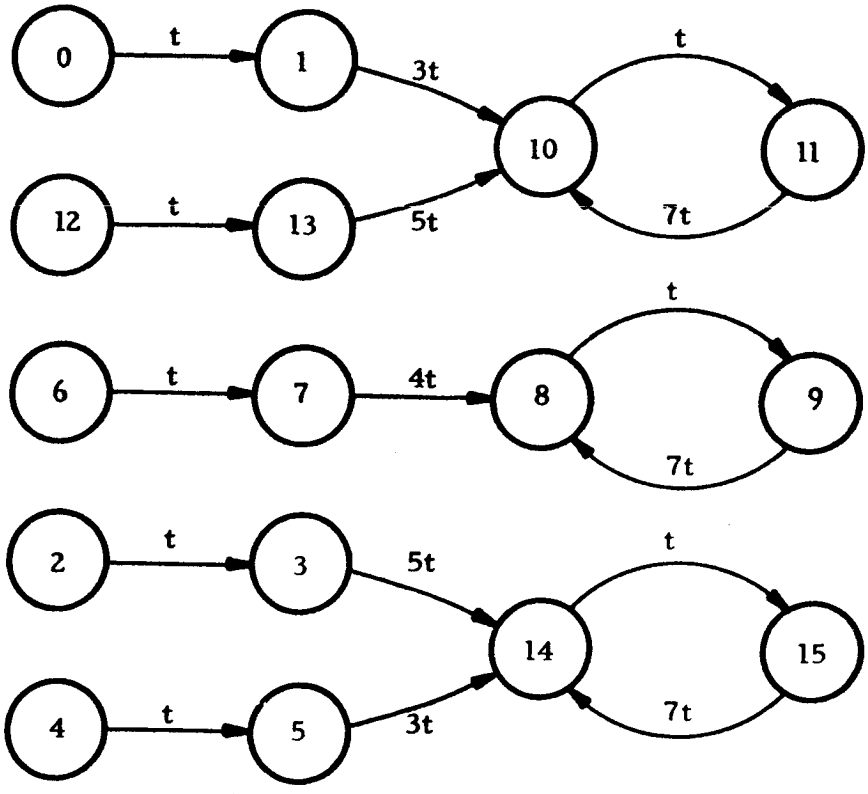


Figure 3.1

State Diagram with Duration of
Circuit Action for Each Transition

obscured the practical aspects of the method's use. At this point, the reader is believed to have sufficient grasp of the technique to inquire into the reasons for the various steps in the technique and for the order in which these steps are undertaken.

In addition, one gap, purposely left in the preceding explanation for the sake of simplicity, will now also be filled in: the treatment of the situation where some of the direct inputs from p are set or reset inputs rather than trigger inputs, giving zeroth transition vectors such as

$$\left[\bar{T}_0\right] = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \bar{A} \end{pmatrix} \qquad (3.16)$$

As will be seen, the disposition of this case is virtually as simple as that of the trigger inputs alone.

Consider, to begin with, a typical transition vector $\left[\bar{T}_k\right]$, resulting from a $\left[\bar{T}_0\right]$ produced by trigger inputs k subintervals ago and propagated through the connective circuitry characterized by $\left[C\right]^t$. Single out a typical element in $\left[\bar{T}_k\right]$, such as $A\underline{\bar{B}\bar{C}D}$ in the Δ_C position. By the notational shorthand previously explained, this term means that stage C will undergo

a transition if A was "1" one subinterval ago and the last stable state, defined at Ot, had a $\bar{B}\bar{C}D$. The difficulty, of course, lies in the determination of A one subinterval ago, and it is quite likely that the engineer studying the circuit will not have this value at hand when consulting the chart; instead, the value of A at Ot will be known. Therefore, it will be convenient at this point to express the unknown value of A in terms of the known value defined at Ot. This will be very simple if such adjustment has been performed for all previous vectors as soon as they were formed; in such a case, all terms in vectors $[T_0]$ through $[T_{k-1}]$ are referred exclusively to the stable state at Ot and the circuit action is easy to follow.

The method for adjusting A in $[T_k]$ is simply based on the reasoning that

$A = \underline{A}$ if stage A underwent an even number of transitions among event times Ot through (k-2)t.

$A = \bar{\underline{A}}$ if stage A underwent an odd number of transitions among event times Ot through (k-2)t.

However, the fact that the literal A without underline appears in a term of $[T_k]$ indicates one of two things:

a. Stage A underwent a transition at event time (k-1)t as well as at any time among (0t) and (k-2)t, and thus produced a transition pulse at kt.

b. Stage A had a level line gated with a transition pulse of other origin produced at (k-1)t. Then A did not undergo a transition at (k-1)t. But, the same effect (i.e., the same present value of A) can be obtained by pretending A was of opposite polarity one subinterval ago and that it underwent a change. If this is done at each step the proper value of A will always be had. The easiest way of carrying out this "pretense" is to modify the source of the A-level term. This is, of course, the connection matrix $[C]^t$. There, the level quantities gated with pulses appear without the superscript t. The same is true for quantities corresponding to S and R inputs (e.g., $A\bar{B}^t$ used to indicate a reset for stage A by an α_B pulse). The modification consists simply of backdating the level quantity one subinterval and complementing it. The result will give all quantities in

$[C]^t$ a superscript of t, which is left off when the matrix is written in the shorthand form for analysis. This explains the mysterious operation described in step 1 on page 74.

In conclusion, then, no matter how the literal A without underline came to be in $[T_k]$, it can always be made to appear as though it came as a result of a transition on the part of stage A at event time (k-1)t -- whether in fact or not. In that case, this transition can be added to those among event times 0t through (k-2)t, giving the following rule for adjusting A in $[T_k]$:

A = $\underline{A}$ if stage A underwent an odd number of transitions among event times 0t through (k-1)t

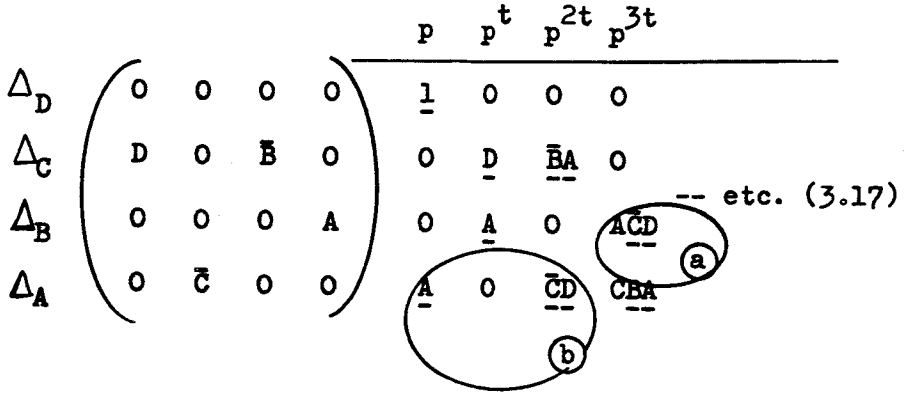
A = $\overline{\underline{A}}$ if stage A underwent an even number of transitions among event times 0t through (k-1)t

By this method all quantities without underlining that appear at kt are easily converted to their underlined counterparts by an examination of vectors $[T_0]$ through $[T_{k-1}]$. Note that all non-underlined quantities now automatically refer to event time kt while all underlined quantities refer

to event time Ot. In short, all explicit reference to other event times has been eliminated. In addition, one simple uniform rule for adjustment of non-underlined literals applies in all cases.

The question still remains, of course, as to how one decides whether or not stage A underwent a transition at some past event time. This is determined by the starting state of the circuit, implied by the underlined portion of the term to be adjusted. Once the terms of a transition vector for a given event time have been expressed in underlined literals only, the Δ_A item in the vector will indicate directly or by implication all starting states for which A will undergo a transition at the event time in question. Thus, in adjusting the non-underlined A, one simply counts among all past vectors the Δ_A terms that imply the starting state, and then complements on an even parity basis.

The problem now arises as to what constitutes implication when a set or reset input was used at time Ot. Specifically, suppose for example a particular circuit is described as follows:



The $\underline{A}$ in the zeroth transition vector indicates stage A was reset by p. The problem now is to adjust A in circled term a. The rule on page 89 states that if an even number of terms in circled area b implies $\underline{\bar{C}D}$, the A should be complemented and underlined; otherwise, it should merely be underlined. However, it is not clear whether or not the term "$\underline{A}$" implies $\underline{\bar{C}D}$. The answer is that if $\underline{A} = 1$, it does; if $\underline{A} = 0$, it does not. Thus, the A in the circled term (a) becomes $\underline{\bar{A}}$ if $\underline{A} = 1$ and $\underline{A}$ if $\underline{A} = 0$. In logical terms this is

$$A = \underline{A\bar{A}} + \underline{\bar{A}A} = 0 \qquad (3.18)$$

Similarly if the zero in circled area (b) were replaced by a $\underline{D}$, say, then the parity relationship for A would change and A would be $\underline{A}$ if A = 1, or $\underline{\bar{A}}$ if $\underline{A} = 0$. This is written as

$$A = \underline{AA} + \underline{\bar{A}\bar{A}} = 1 \qquad (3.19)$$

This interesting result bears out, of course, the fact

that A can under no circumstances be equal to 1 at event time 2t since it was reset at Ot and did not change at t. However, the result was obtained without recourse to intuitive reasoning.

Setting up equations such as (3.18) and (3.19) is not only confusing but unnecessary. The same result is obtained by treating the A in the zeroth vector as 1 and then resetting A in $A\underline{\underline{\bar{C}D}}$ if circled area b has an even number of terms implying $\underline{\underline{\bar{C}D}}$ or setting A otherwise.

Clearly, for the example in (3.17) 1 implies $\underline{\underline{\bar{C}D}}$, giving $A\underline{\underline{\bar{C}D}} = 0$.

The identical reasoning applies to set inputs. If stage A had been set instead of reset at time Ot, an $\underline{\bar{A}}$ would replace $\underline{A}$ in $[T_0]$. Then, to adjust A in $A\underline{\underline{\bar{C}D}}$, again treat $\underline{A}$ as 1, which here simply means treat $\underline{\bar{A}}$ as 0, and repeat the procedure described for resets. The result for the example in (3.17) would have been $A\underline{\underline{\bar{C}D}} = \underline{\underline{\bar{C}D}}$.

To sum up the techniques for treating set and reset inputs, treat both exactly the same: in $[T_0]$, replace $\underline{Q}$ in Δ_Q by 1; check parity of transitions as for trigger inputs; reset on even parity, set on odd parity.

Logic Reduction of Transition Vectors

Relatively simple logical expressions appeared in the transition vectors used for examples in the past section and

the question of logic reduction did not enter the discussion. Yet, since simplicity is a keynote of the subinterval analysis technique, such an important avenue to its achievement must certainly be explored.

The terms of a particular transition vector will be of some general logical form, not necessarily reduced with respect to any common criterion for simplicity. Most often, this will be the union-of-intersections (sum-of-products) form. Before hastily undertaking to simplify such terms by any of the popular reduction methods, one must first realize that certain algebraic peculiarities enter into the reduction of logical quantities having mixed event time references. Secondly, one should be aware that the unreduced logical expression for a transition may contain important information that vanishes with logic reduction. Specifically, the particular constitution of a transition signal may indicate the presence of a pathological timing condition, such as a critical R-S, R-T, or S-T race, arising from an unplanned and unwelcome signal coincidence.

Figure 3.2 shows a circuit with a critical R-T race at stage A for a starting state 001. To uncover this fact, examine the subinterval analysis for the circuit:

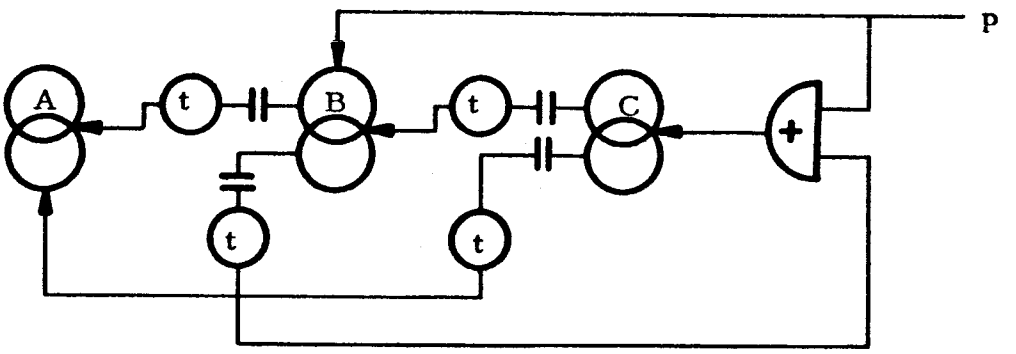


Figure 3.2

$$
\begin{array}{cccccccc}
 & & & & p & p^t & p^{2t} & \\
\Delta_C & \Big(& 0 & \bar{B} & 0 & 1 & \bar{B} & \bar{B}\bar{C} = 0 \\
\Delta_B & & C & 0 & 0 & \bar{B} & C & C\bar{B} = \bar{C}\bar{B} \longrightarrow \\
\Delta_A & & \bar{A}\bar{C} & B & 0 & 0 & A\bar{C} & \bar{A}\bar{C}\bar{B} + B\bar{C} = \textcircled{A\bar{C}\bar{B} + \bar{C}} \\
 & & & & & & & (3.20)
\end{array}
$$

The encircled term is a warning of possible trouble. Yet were it reduced simply to $\bar{C}$ according to the usual rules of logic simplification, the warning would be lost. Specifically, the term is composed of two quantities having a common intersection. This in itself would not constitute a problem in the case of a T memory unit with an OR gate at the input providing the individual signals were sufficiently close to be unresolvable to the memory unit. This case, however, involves an RT memory and the two distinct terms $A\bar{C}\bar{B}$ and $\bar{C}$ indicate that for a starting state of $A\bar{C}\bar{B}$ at least, both a reset and a trigger pulse will be applied to stage A. Disregarding for a moment the fact that these pulses are required to be non-coincident by stipulation, the most optimistic statement that could be made concerning this situation would be that if stage A is in the "1" state, the R and the T commands are not in conflict. If, on the other hand, stage A is in the "0" state, the two coincident pulses are in opposition; this constitutes a critical race with the

final outcome depending on which signal arrives at its destination first and by how much.

To identify the starting state or states producing a critical race at this point, if any, a very simple technique is used. It is based on the following line of reasoning:

To begin with, the circled terms $\underline{AC\bar{B}}$ and $\underline{C}$ in (3.20) identify transitions, not input signals. Now for $\underline{C}$, associated with a trigger input, the two are synonymous since the memory unit will switch whenever triggered. The shortcoming is that the polarity of the transition is not known. In contrast, the polarity of the transition, if any, associated with $\underline{AC\bar{B}}$ is known (it can only be a β transition), but what is not known is all the conditions for which a pulse will appear at the R input of stage A. The condition being sought, of course, is the appearance of a pulse at the R input and simultaneously, a required α transition initiated by a pulse on the T input. In other words, the information needed is precisely what is not known about both inputs.

The problem is resolved in straightforward fashion, without the necessity of obtaining a general expression for A at event time 2t, (relatively easy for this example but often very laborious).

Consider the transition vector $[T_1]$ and the circuit action defined by $[C]^t$ that eventually produces $[T_2]$. If one could somehow reach into the circuit at event time t and quickly move the trigger input line of stage A to its set input and its reset input line to its trigger input, the term replacing the trigger term $\underline{C}$ would indicate only that subset of $\underline{C}$ which results in an α transition. The former reset input, on the other hand, is now a trigger input so the former reset term $\underline{AC\bar{B}}$ would be replaced by one indicating all starting conditions for which a pulse appears on this line. The two new terms thus give precisely the information sought and their intersection specifies directly the starting states for which a critical race occurs at event time 2t.

The input line manipulation just mentioned is easily performed mathematically by picturing a temporary modification of $[C]^t$. For the example in (3.20), imagine that the Δ_A (third) row of the transition matrix is at event time t momentarily changed to

$$\Delta_A \ (\bar{C} \quad AB \quad 0) \tag{3.21}$$

The resulting Δ_A term in $[T_2]$ is then

$$\underline{\bar{C}\bar{B}} + \underline{ABC} = \underline{C\bar{B}} + \underline{\bar{A}C} \tag{3.22}$$

The intersection of the two terms is $\underline{\bar{A}\bar{B}C}$, identifying 001

Consider the transition vector $[T_1]$ and the circuit action defined by $[C]^t$ that eventually produces $[T_2]$. If one could somehow reach into the circuit at event time t and quickly move the trigger input line of stage A to its set input and its reset input line to its trigger input, the term replacing the trigger term $\underline{C}$ would indicate only that subset of $\underline{C}$ which results in an α transition. The former reset input, on the other hand, is now a trigger input so the former reset term $\underline{AC\bar{B}}$ would be replaced by one indicating all starting conditions for which a pulse appears on this line. The two new terms thus give precisely the information sought and their intersection specifies directly the starting states for which a critical race occurs at event time 2t.

The input line manipulation just mentioned is easily performed mathematically by picturing a temporary modification of $[C]^t$. For the example in (3.20), imagine that the Δ_A (third) row of the transition matrix is at event time t momentarily changed to

$$\Delta_A \ (\bar{C} \quad AB \quad 0) \qquad (3.21)$$

The resulting Δ_A term in $[T_2]$ is then

$$\underline{\bar{C}\bar{B}} + \underline{ABC} = \underline{C\bar{B}} + \underline{\bar{A}C} \qquad (3.22)$$

The intersection of the two terms is $\underline{\bar{A}\bar{B}C}$, identifying 001

as the starting state for which there is a critical race at 2t.

Thus, if only critical races are to be avoided, the 001 state must be avoided, or the circuit must be redesigned. If all R-T coincidence is to be avoided, as is more commonly the case, the starting conditions involved are defined by the intersection of $\underline{C}$ (identifying all trigger inputs) and $\underline{C\bar{B}}$ (identifying all reset inputs); namely, $\underline{C\bar{B}}$, corresponding to starting states 001 and 101.

The operation just discussed is readily generalized for R-S-T memory units. To avoid all input coincidence, at a given event time all inputs are imagined to be trigger inputs at the preceding event time and the trouble-producing conditions are found by multiplying the former R, S, and T quantities among themselves, two at a time, and then taking their union. For example, a modified Δ_A term $\underline{A\bar{B}} + \underline{A\bar{C}} + \underline{B\bar{C}} = x + y + z$ would identify the trouble-producing starting condition $xy + xz + yz = \underline{A\bar{B}\bar{C}} + \underline{AB\bar{C}} = \underline{AC}$. In obtaining the above result, it is important to have the individual terms x, y, and z maintain their separate identities until one is ready to combine them, since they may be more complicated than simple product terms. The easiest way to do this is to separate clearly each quantity arising from a single term of the matrix $\left[C\right]^t$.

If only critical races are to be avoided for an RST memory, the inputs must be examined two at a time. Eliminate R-T critical races by repeating the techniques described on page 97. Repeat the process for S-T critical races by temporarily replacing T by R and S by T. Finally eliminate all R-S coincidences -- since they always constitute a critical race -- by temporarily replacing both R and S by T. Each such examination will identify the starting conditions producing the particular race. The union of these three sets of conditions identifies the starting conditions producing a critical race at the event time in question.

Once all timing problems for a given transition vector have been identified and resolved, logic reduction of the individual components can be undertaken without further concern. The usual rules of logic reduction apply, providing all literals in a component are underlined. They do not apply when underlined and non-underlined literals are mixed. For example, the operation

$$\underline{A\bar{B}CD} + \underline{ABCD} = A(\underline{BCD} + \underline{\bar{B}CD}) = \underline{ACD} \qquad (3.23)$$

is not legitimate simply because the value of $\underline{A}$ corresponding to each A may not be the same. An example of such a case is

shown below:

	p	p^t	p^{2t}	p^{3t}
Δ_B	-----	-----	-----	$\underline{ABCD} + \underline{A\bar{B}CD}$
Δ_A	0	$\underline{CD}$	$\underline{\bar{B}CD}$	-----

(3.24)

Now notice that only $\underline{CD}$ (an odd number of terms) implies BCD, while both $\underline{CD}$ and $\underline{\bar{B}CD}$ (an even number of terms) imply $\underline{\bar{B}CD}$. Therefore, the A in $\underline{ABCD}$ is simply underlined while in $\underline{A\bar{B}CD}$, it is complemented and underlined. The result is

$$\underline{ABCD} + \underline{A\bar{B}CD} = \underline{ABCD} + \underline{\bar{A}\bar{B}CD} = \underline{CD}\,(\underline{A} \oplus \underline{\bar{B}}) \tag{3.25}$$

The general rule for simplification of mixed terms is that <u>non-underlined factors of the same underlined product ($\underline{\mathcal{A}}$) can be collected and simplified, but the converse is not true</u>. For example, the operation

$$\underline{ABCD} + \underline{\bar{A}BCD} = (A + \bar{A})\underline{BCD} = \underline{BCD} \tag{3.26}$$

is valid since both terms are based on the same starting condition $\underline{BCD}$. This makes the value of $\underline{A}$ the same for both terms allowing the use of the relation $A + \bar{A} = 1$.

The general rule just stated has been given mainly for the sake of completeness. In practice, an operation such

as (3.26) should be avoided because it will eliminate possible indications of critical races before they can be identified and analyzed. It is far preferable to convert all mixed terms to their fully underlined equivalents without simplification. All reduction previously possible will still be possible in the fully underlined form, after all critical races have been eliminated.

This concludes the development of the subinterval analysis technique to which this chapter has been exclusively devoted. Throughout this development, the aim has been the formulation of a unified algorithm having an optimally small set of simple and uniform rules.

In order to provide this simplicity and uniformity, certain modifications in the actual circuit were at times mathematically simulated in order to fit within the carefully selected rules and still furnish correct results. The correct results, it is felt, justify the use of these hypothetical modifications. They can be dispensed with, on the other hand, only at the cost of a much more extensive set of rules and a corresponding increase in bookkeeping complexity.

IV. SYNTHESIS OF TRANSITION-COUPLED COUNTERS

In Chapters II and III, a logic algebra was formulated within whose framework a transition-coupled asynchronous counter of known structure can be described in a consistent logical-algebraic form. Using such a description as a working basis, analysis techniques were then developed by means of which the general stability and performance characteristics of a circuit can be methodically investigated.

Throughout the discussion thus far, the existence of the counter under consideration was assumed a priori and at no time did the question arise as to how the counter was conceived and why it was given its particular structure.

The surprising fact is that the overwhelming majority of transition-coupled circuits are designed by heuristic means alone. Their frequent simplicity and reliability attest to the ingenuity and creativeness of their designers.

The reason intuition plays a major role in the design of transition-coupled circuits is simply that few tangible design criteria or operation specifications are available for the general circuit. All that is often specified is the sequence of stable states through which the circuit is expected to progress during its operation. Hence, the choice of possible

unstable states assumed during transition, coupled with that of logic circuitry based on some suitably defined minimum cost criterion, constitutes a degree of freedom that is appalling.

Nevertheless, the literature has not ignored the problem. In particular, the transition-coupled circuit in which any memory unit can undergo no more than one transition between stable states (see Chapter II) constitutes a special case that lends itself to moderately straightforward design procedures. Mergler [24] and Marcus [21], for example, have developed adaptations of the Karnaugh map design method, using transitions as additional variables, but only Mergler has used them as actual line signals. Eichelberger [11] has also considered this case at length and has developed a new type of circuit realization called a "delayed-input circuit". This circuit, inherently free of critical races, generates "hazard pulses" (pulses generated by deliberate delays) to change its internal state and is in actuality not a "coupled" circuit in its truest sense; that is, memory unit transitions are accomplished independent of each other.

Once the restriction of this special type of operation is removed, the design problem is beyond the scope of formal optimization techniques. To aggravate matters, extensions of

design procedures available for special cases give extremely poor results when applied to the general case -- hence, the need for intuitive design techniques by the engineer.

This chapter will attempt to facilitate the intuitive design approach by furnishing a formal basis about which the design can evolve. Examples of actual designs will also be presented.

One of the principal problems in synthesizing transition-coupled counters is that of obtaining a good preliminary description of the circuit's action. Specifically, the common systems of circuit specification describe a circuit in terms of its level-coupled counterpart. What is needed in its stead is a form of description that can describe a transition between stable states in a way that clearly suggests asynchronous transition coupling among the individual stages. To be sure, the description could not be expected to yield <u>all</u> the possible coupling schemes for a given counter requirement, but it might be expected to indicate <u>one</u> of them.

The easiest way of obtaining a description such as mentioned above is to envision the action of a counter to be designed in terms of equivalent action by a transition-coupled counter already known. The familiar counter thus serves as a "model" for the one being synthesized and as the count sequence

for the latter is studied, the question is asked, "What inputs and added coupling would be required to make the model describe the required sequence?" If these inputs and additions prove to be feasible, the required counter is then designed as an adaptation of the model.

The "model" counter to be used as an example in this chapter is the familiar <u>standard binary adder</u>, shown in Figure 4.1. It consists simply of a set of T-type memory units with each trigger input connected to the p^t output of its neighbor to the right and to a possible p input from the outside. The connection matrix $[C]^t$ and zeroth vector $[T_Q]$ for the circuit as shown in Figure 4.1 are

$$\begin{array}{c} \\ \Delta_D \\ \Delta_C \\ \Delta_B \\ \Delta_A \end{array} \begin{pmatrix} 0 & 0 & 0 & 0 \\ D & 0 & 0 & 0 \\ 0 & C & 0 & 0 \\ 0 & 0 & B & 0 \end{pmatrix} \begin{array}{c} p \quad p^t \\ \hline \underline{a}_0 \\ \underline{a}_1 \\ \underline{a}_2 \\ \underline{a}_3 \end{array} \text{----- etc.} \qquad (4.1)$$

The feature of particular interest, however, is that if the binary word ABCD is read as a number in the standard binary code having the numerical value

$$V = 8A + 4B + 2C + D \qquad (4.2)$$

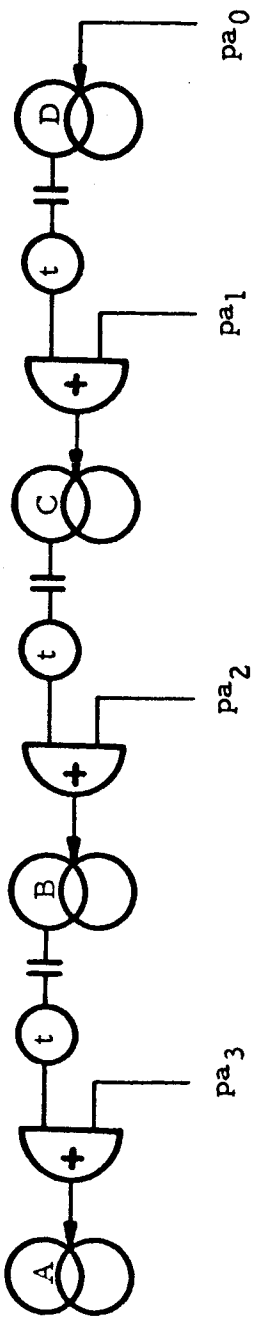


Figure 4.1

then the circuit shown adds the number

$$v = 8a_3 + 4a_2 + 2a_1 + a_0 \qquad (4.3)$$

to the contents of the counter. The resulting state of the counter is the sum modulo 16 of V and v.

It should be noted that the symbol + in (4.2) and (4.3) is the standard algebraic addition (PLUS) connective rather than the logical union (OR) connective.

From equations (4.2) and (4.3) it is apparent that the behavior of the binary adder can be described in a very compact form. Using the relation

$$(V' - V) \bmod 16 = v \qquad (4.4)$$

the change in the reading of the circuit contents can easily and uniquely be expressed as the numerical value of v. For example, for $a_0 = 1$, $a_1 = a_2 = a_3 = 0$, the equation fully specifying the counting action of the circuit in Figure 4.1 is

$$v = 1 \qquad (4.5)$$

In verbal terms, this simply states, "To go from any stable state to the next stable state, add 1."

It is now only a short step to the extension of the above philosophy to include sequences other than those characterized

by the addition of a constant. Consider, for example, a general cyclic sequence of three-digit words

$$\begin{array}{c} \vdots \\ (A_0 \quad B_0 \quad C_0) \\ (A_1 \quad B_1 \quad C_1) \\ \vdots \\ (A_7 \quad B_7 \quad C_7) \\ (A_0 \quad B_0 \quad C_0) \\ \vdots \end{array} \qquad (4.6)$$

Assume for the moment that there is no redundancy (i.e., all eight possible words are permissible) or ambiguity (i.e.,the sequence is cyclic in excctly 8 words). Then, if each binary word is assigned a numerical value according to

$$V_k = 4A_k + 2B_k + C_k \qquad (4.7)$$

the transition between any pair of successive words can be characterized by the addition of a numerical quantity given by

$$v_k = (V_{k+1} - V_k) \bmod 8 \qquad (4.8)$$

Thus, a value of v_k corresponds to every value of V_k in the sequence. For the general case, v_k is not a constant (as in the case of the standard binary adder) but is a function of the particular value V_k

$$v_k = v_k(V_k) \tag{4.9}$$

and as a result of (4.7) this naturally makes v_k a function of the digits A_k, B_k and C_k.

$$v_k = v_k(A_k, B_k, C_k) \tag{4.10}$$

The general form for v is

$$v = \bar{A}\bar{B}\bar{C}\, v(0,0,0) + \bar{A}\bar{B}C\, v(0,0,1) \ldots\ldots$$
$$\ldots\ldots + ABC\, v(1,1,1) \tag{4.11}$$

This interesting sum of <u>weighted minterms</u> is simply a formal standard-algebraic statement replacing a tabular list of relations in the form of (4.10). Whereas an expansion of this type is informative, one of more interest for purposes of this discussion is the equivalent <u>weighted exterm [13] expansion</u>:

$$v = w_0 + w_1C + w_2B + w_3BC + w_4A$$
$$+ w_5AC + w_6AB + w_7ABC \tag{4.12}$$

Since (4.12) must fit 8 distinct values of v and corresponding values of A, B, and C, for the non-redundant case, the values of the w coefficients are fully and uniquely determined. They are the solution to the set of linear equations

$$v(V_k) = w_0 + w_1 C_k + \ldots + w_7 A_k B_k C_k \qquad (4.13)$$

or

$$[v] = [E][w] \qquad (4.14)$$

where $[v]$ and $[w]$ are column vectors, $[E]$ is a square Boolean matrix representing the exterm composition for each specific word, and the matrix multiplication is standard algebraic. For the non-redundant case $[E]$ is non-singular (its determinant is equal precisely to 1), giving a unique set of w's as a solution. A convenient way of expressing this solution is

$$\begin{aligned}
w_0 &= v(0) \\
w_1 &= v(1) - w_0 \\
w_2 &= v(2) - w_0 \\
w_4 &= v(4) - w_0 \\
w_3 &= v(3) - w_2 - w_1 - w_0 \\
w_5 &= v(5) - w_4 - w_1 - w_0 \\
w_6 &= v(6) - w_4 - w_2 - w_0 \\
w_7 &= v(7) - \sum_{j=0}^{6} w_j
\end{aligned} \qquad (4.15)$$

This set of equations points out the simple procedure for determining the w coefficients for a non-redundant sequence from a listing of the words and their corresponding values of v. One begins by finding w_0, which is simply the value of v opposite the word (000). Using this value, coefficients corresponding to single-literal exterms (A, B, etc.) are found by consulting v values for words containing single 1's only. The results are then applied to words containing two 1's and so forth, until all the values of w have been found.

The procedure just described can best be illustrated with an example. Table 4.1 shows a desired sequence with the values of v shown and also the composition of v in terms of w. The sequence will be recognized as the three right-most digits of the four-bit Parity Checked Gray (PCG) Code. The values of v are shown in positive mod 8 form, although negative values are equally acceptable from a formalistic point of view. The values of the w coefficients are now easily found:

Row	A B C	v	Composition
①	0 0 0	3	w_0
②	0 1 1	3	$w_0 + w_1 + w_2 + w_3$
③	1 1 0	7	$w_0 + w_2 + w_4 + w_6$
④	1 0 1	7	$w_0 + w_1 + w_4 + w_5$
⑤	1 0 0	3	$w_0 + w_4$
⑥	1 1 1	3	$\sum_{j=0}^{7} w_j$
⑦	0 1 0	7	$w_0 + w_2$
⑧	0 0 1	7	$w_0 + w_1$

Table 4.1

$$
\begin{aligned}
&\text{From } ① \quad w_0 = 3 \\
&\text{"} \quad ⑦ \quad w_2 = 4 \\
&\text{"} \quad ⑧ \quad w_1 = 4 \\
&\text{"} \quad ⑤ \quad w_4 = 0 \\
&\text{"} \quad ② \quad w_3 = 0 \\
&\text{"} \quad ③ \quad w_6 = 0 \\
&\text{"} \quad ④ \quad w_5 = 0 \\
&\text{"} \quad ⑥ \quad w_7 = 0
\end{aligned}
\tag{4.16}
$$

The result indicates that the PCG counter is characterized by the arithmetic equation

$$v = 3 + 4C + 4B \tag{4.17}$$

What this means in verbal terms is simple "To go to the next stable state, perform the addition operation indicated by (4.17) on the contents of the counter."

As to how such addition is performed, it will be recalled that any numerical value is <u>added</u> by triggering simultaneously the stages corresponding to the location of 1's in the standard binary representation of the addend. Thus, 3 is added by triggering stages B and C, and 4 is added by triggering stage A.

Notice also that in adding 3 by triggering stages B and C, a resulting β_B pulse will indicate a B at time Ot, and a β_C pulse will indicate a C at time Ot. Thus, pulsed signals

representing B and C are available as a by-product of the addition of 3 and conceivably they could be stored by means of delays and eventually applied to stage A as additional addends. Unfortunately, the matter is not as simple as it might appear. First of all, the delayed signals would have to be kept from interfering not only with the propagation of the usual carry signals associated with the addition of 3, but also with each other. Secondly, to aggravate matters, B is not the only logical quantity producing a β_B signal. A brief analysis using the subinterval analysis technique would disclose that a β_B can also appear at event time 2t as a result of $\bar{B}C$ at 0t. In short, a direct implementation of equation (4.17) is not easily and practicably achieved. However, equation (4.17) is a standard algebraic equation and can thus be manipulated according to standard algebraic rules. Such a manipulation results in a rather convenient synthesis, as will now be shown.

To begin with, all quantities in (4.17) are modulo 8; therefore +4C can be replaced by -4C, which in turn can be written as -3C-C. Equation (4.17) then becomes

$$v = 3 - 3C - C + 4B \qquad (4.18)$$

$$= 3(1-C) - C + 4B \qquad (4.19)$$

But (1-C) is the standard-algebraic expression for $\bar{C}$, which finally yields

$$v = 3\bar{C} - C + 4B \qquad (4.20)$$

The implementation of this equation is very simple in contrast to (4.17). The term "-C" means, "If C=1, complement (reset) it without carry." The "$3\bar{C}$" indicates, "If C = 0, complement (set) it and use the α_C pulse thus generated as a trigger input for stage B. (The setting of stage C adds $\bar{C}$, while the triggering of stage B adds 2 whenever there is a $\bar{C}$, i.e. adds $2\bar{C}$)."

What this reasoning requires in practical terms is simply to remove the usual $\beta_C \longrightarrow T_B$ line of the standard binary adder and introduce instead a $\alpha_C \longrightarrow T_B$ line. Simply triggering stage C then accomplishes the $3\bar{C}$ - C operation. As for the addition 4B, it is best to dispose of this at time Ot, when there are no timing problems. Namely, no transition signals appear at any input of stages A or B at this time, making it a simple matter to trigger stage A with the pulsed signal pB, formed by means of an AND gate on the "1" level output of stage B. The subinterval equations for the resulting circuit are

$$\begin{aligned} T_C &= p \\ T_B &= \alpha_C^t \\ T_A &= pB + \beta_B^t \end{aligned} \qquad (4.21)$$

The synthesized circuit is shown in Figure 4.2. Its stability is assured by the absence of loops.

The subinterval analysis for the circuit appears below.

$$\begin{array}{c|ccc|cccc} & & & & p & p^t & p^{2t} & p^{3t} \\ \hline \Delta_C & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \Delta_B & \bar{C} & 0 & 0 & 0 & \underline{\bar{C}} & 0 & 0 \\ \Delta_A & 0 & B & 0 & \underline{B} & 0 & \underline{B\bar{C}} & 0 \end{array} \qquad (4.22)$$

From (4.22), the next states in terms of the present states are given by

$$\begin{aligned} C' &= C \oplus 1 = \bar{C} \\ B' &= B \oplus \bar{C} \\ A' &= A \oplus B \oplus B\bar{C} = A \oplus BC \end{aligned} \qquad (4.23)$$

These equations are correct for the PCG code, as may be easily ascertained by a Karnaugh map or any other standard method.

Several points should be particularly noted in the synthesis just completed. To begin with, the circuit shown in Figure 4.2 is an adaptation of a well-known transition-coupled circuit -- the standard binary adder -- and is thus itself a transition-coupled circuit. Secondly, the circuit has only one gate in addition to input buffering: a 2-legged

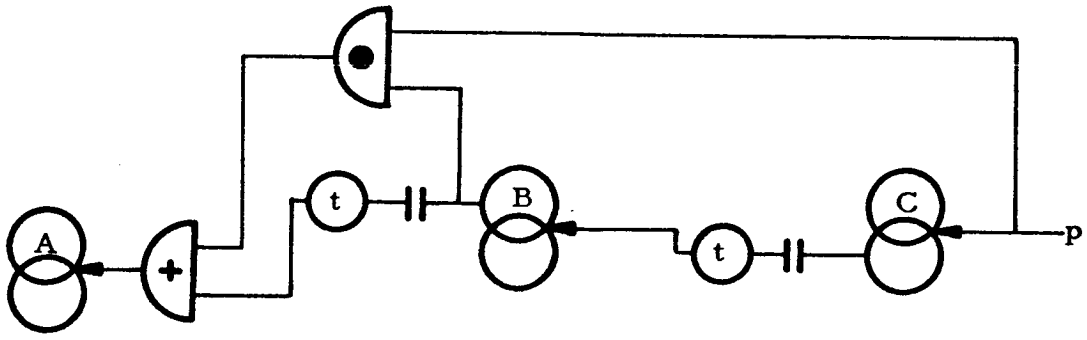


Figure 4.2

AND gate. This is a considerably less costly design than would be required for an equivalent synchronous counter free of hazards and critical races.

Thirdly, it will be noticed that the counter has no feedback loops; all signals travel "upstream". This follows from the fact that the PCG code is a minimum dependency code [3] and is not characteristic of all codes. The feature does indicate, however, that the relative location of the digit positions for comparison with a standard binary adder is a matter of some consequence. Had any other arrangement of digit positions been chosen (say, $V = 4C + 2A + B$), the arithmetic analysis would have suggested an entirely different synthesis from the one just completed. The problem of digit position placement will be discussed further shortly.

As a final comment on the circuit shown in Figure 4.2, notice that if stage C is "turned upside down" (i.e., the $\bar{C}$ side is regarded as the "1" side, α_C as $\beta_{\bar{C}}$, β_C as $\alpha_{\bar{C}}$ and V is defined as $4A + 2B + \bar{C}$) then the circuit takes on again the appearance of a standard binary adder. In other words, had the entire synthesis problem been considered in terms of generating a sequence of numbers $(A_k, B_k, \bar{C}_k)$, rather than (A_k, B_k, C_k), the arithmetic manipulation of (4.17)

to produce (4.20) would have been unnecessary. The same result would appear directly as

$$v = 1 + 4B \qquad (4.24)$$

This fact points out that in addition to relative digit position placement, the choice as to which side of each stage to use as the "1" side in the binary adder analogy also plays a major role in obtaining a simple and elegant synthesis.

From the preceding comments, one may begin to appreciate the magnitude of the degree of freedom facing the circuit designer even in relatively elementary synthesis problems. To begin with, there are n! distinct ways of ordering n digit positions. Secondly, there are 2^n ways of selecting the "1" sides for the n stages to be used (any of the 2^n possible words can be regarded as the 00 .....00 word).

Finally, if the case is redundant, i.e., $r(<2^n)$ words never occur in the sequence, then there is in addition a set of r equations of the form

$$m_j (A,B,C....) = 0 \qquad (4.25)$$

where m_j is the minterm corresponding to a redundant word. Since the redundant words never occur, any value of v may be

specified for each of them, providing no error checking or correcting is required. In the mod 2^n system, this amounts to 2^n possible values of v for each redundant word, any of which -- not necessarily zero -- may result in a simplification of the final synthesis.

It is an unfortunate fact that to date, no clear-cut algorithm has been found that will indicate the optimum choice of digit positions, variable assignments, and v values for redundant combinations, where applicable. Furthermore, even when one choice among all those mentioned has been made and the corresponding synthesis equation has been obtained, by far the simplest way of obtaining the synthesis equation for some other assignment of digit positions, variables, etc., is to repeat the entire process used to find the first equation. With few exceptions, conversion of existing results from one frame of reference to another is difficult, though possible.

Despite the difficulties just discussed, certain considerations prove helpful in finding a promising assignment of digit positions, variables, etc. These will now be briefly examined.

It was shown in reference 3 that the logical dependency among the stages in a comparator (a form of adder) on each

other is minimized if the digit positions are arranged in order of increasing periodic cycle. Since the nature of the logical dependency is not specified in this finding, the result is applicable to cases where such dependency is in the form of transition coupling as well as level coupling.

In setting up positions for a counter synthesis, the positions with the shortest periodic cycle are placed at the right and given a weighting of 1. The other positions are arranged so that the longer the periodic cycle, the higher the weighting. It has been found that where periodicity is the same for a number of stages, the simplest syntheses are obtained if positions having significant portions of short-period cycles (though occasional interruptions increase the overall periodic cycle) are placed to the right of positions not having such cycles.

The position assignment rule just described not only minimizes the dependency distribution among the stages in the counter, but also tends to insure greater stability by minimizing the number of loops needed. This is apparent in the design of the PCG counter shown in Figure 4.2. Position C has a period of 2 counts and was thus placed at the right. Position B has a period of 4 counts and was therefore placed

to the left of C and to the right of A, which has a period of 8 counts. The resulting synthesis was loop-free as was previously pointed out. Had some other ordering of positions been used, the loop-free design possibility would not have been easily discerned from the corresponding synthesis equation.

The choice of variables ("1" sides) for the synthesis problem is considered after the digit positions have been assigned. The chief motivation in this phase of the design problem is avoidance of signal coincidence or critical races throughout the circuit. The existence and number of these timing problems will be determined to a great extent by the value of the constant w_0 which indicates the external trigger excitation if the synthesis equation is implemented directly. Now, if w_0 is a number whose binary representation contains a relatively large number of 1's, then many stages will be triggered by p at event time Ot and a correspondingly large number of transition signals is likely to be generated at this time. This, in turn, makes serious timing problems difficult to avoid, since a definite destination must be found for all signals within a single subinterval. Where

such destination is the same for two or more signals, a temporary storage in the form of a delay is required for one or more of these signals. This will tend to increase component cost and hamper the speed. Therefore, as a starting point, the designer should seek a variable choice by which the binary representation of w_0 has as few 1's as possible, preferably only one. As for its location, the nearer it is to the right-hand end of the word, the better, since this will minimize the need for feedback paths, as previously discussed.

As may be inferred, then, a significant problem is determining quickly and easily the w_0 coefficient for any choice of variables. This is a simple procedure as will now be shown:

For any choice of variables, w_0 is the numerical value of the word following the 00 ..... 00 word, as seen from the frame of reference. Now suppose $(A_k B_k C_k \ldots\ldots)$ is chosen as the 00 .....00 word.

To obtain this frame of reference, every "1" in $(A_k B_k C_k \ldots)$ had to be complemented to produce 0. Then, the same operation must be performed on all the other numbers in the sequence to place them in the same frame of reference. In particular,

the number $(A_{k+1}, B_{k+1}, C_{k+1}\ldots)$ is transformed to the new frame of reference by simply ring summing it with $(A_k, B_k, C_k,..)$, digit by digit. The resulting number is the binary representation of w_0.

Using the above rule, a complete listing of w_0 values is easily obtained by ring summing the successive words of a sequence in pairs, bit by bit. Table 4.2 shows the values of w_0 for all the possible choices of variables for the 3-digit PCG code.

As may be seen from this table, the variable choice (A,B,C) has a w_0 coefficient of 3, as was found in equation set (4.16). Later, it was pointed out that the preferable variable choice $(A,B,\bar{C})$ had associated with it a w_0 of 1. This is seen in the last line of the table. It will also be noticed that the variable choice $(\bar{A},B,\bar{C})$ has a w_0 coefficient of 1 as well. This would seem to indicate that this choice is another promising one for a simple synthesis. The conjecture is borne out by the resulting synthesis equation

$$v = 1 + 4B \qquad (4.26)$$

which is identical to that associated with the variable choice $(A,B,\bar{C})$.

A	B	C	Reference Variables	w_o, binary	w_o, decimal
0	0	0	(A, B, C)	0 1 1	3
0	1	1	(A, $\bar{B}$, $\bar{C}$)	1 0 1	5
1	1	0	($\bar{A}$, $\bar{B}$, C)	0 1 1	3
1	0	1	($\bar{A}$, B, $\bar{C}$)	0 0 1	1
1	0	0	($\bar{A}$, B, C)	0 1 1	3
1	1	1	($\bar{A}$, $\bar{B}$, $\bar{C}$)	1 0 1	5
0	1	0	(A, $\bar{B}$, C)	0 1 1	3
0	0	1	(A, B, $\bar{C}$)	0 0 1	1

Table 4.2

To illustrate the concepts thus far presented, consider the following example:

It is desired to synthesize a three-bit counter that will begin in state (000), go to (001), and through all odd binary numbers, then go back to (010) and proceed through all even numbers, returning to (000) from the last reading of (110). The count sequence is:

A	B	C
0	0	0
0	0	1
0	1	1
1	0	1
1	1	1
0	1	0
1	0	0
1	1	0
0	0	0
	⋮	

(4.27)

First of all, notice that position C is the most regular of the positions and has no low-period cycles. Therefore, it should be placed in the leftmost location. The distinction between positions A and B is somewhat less well defined but

position A does appear to be more "regular" of the two, as it contains shorter segments of regularly alternating 1's and 0's and undergoes only 4 transitions in the full cycle compared to 6 for position B. Therefore, the position assignment will be CAB.

Next, the variable choice must be considered. The w_0 values are listed in Table 4.3.

There are 3 choices -- 2 , 4 , and 7 -- for which $w_0 = 1$. Choice 2 will be adopted for this example.

The desired sequence is now written in the new frame of reference (i.e., with column C complemented, giving $V = 4\bar{C}+2A+B$) and the value of v for each word is entered next to the word. This is shown in Table 4.4.

The w coefficients are now found as in the previous example:

$$\begin{aligned}
&\text{From } (2) \quad w_0 = 1 \\
&\text{"} \quad (3) \quad w_1 = 0 \\
&\text{"} \quad (4) \quad w_2 = 0 \\
&\text{"} \quad (1) \quad w_4 = 3 \\
&\text{"} \quad (5) \quad w_3 = 1 \\
&\text{"} \quad (6) \quad w_5 = 5 \\
&\text{"} \quad (7) \quad w_6 = 5 \\
&\text{"} \quad (8) \quad w_7 = 6
\end{aligned} \tag{4.28}$$

Choice	C	A	B	Reference Variables	w_0
①	0	0	0	(C, A, B)	4
②	1	0	0	($\bar{C}$, A, B)	1
③	1	0	1	($\bar{C}$, A, $\bar{B}$)	3
④	1	1	0	($\bar{C}$, $\bar{A}$, B)	1
⑤	1	1	1	($\bar{C}$, $\bar{A}$, $\bar{B}$)	6
⑥	0	0	1	(C, A, $\bar{B}$)	3
⑦	0	1	0	(C, $\bar{A}$, B)	1
⑧	0	1	1	(C, $\bar{A}$, $\bar{B}$)	3
	0	0	0		

Table 4.3

Row	$\bar{C}$ A B	v	Composition
①	1 0 0	4	$w_0 + w_4$
②	0 0 0	1	w_0
③	0 0 1	1	$w_0 + w_1$
④	0 1 0	1	$w_0 + w_2$
⑤	0 1 1	2	$w_0 + w_1 + w_2 + w_3$
⑥	1 0 1	1	$w_0 + w_1 + w_4 + w_5$
⑦	1 1 0	1	$w_0 + w_2 + w_4 + w_6$
⑧	1 1 1	5	$\sum_{j=0}^{7} w_j$

Table 4.4

The basic synthesis equation is therefore

$$v = 1 + 3\bar{C} + AB + 5\bar{C}B + 5\bar{C}A + 6\bar{C}BA \qquad (4.29)$$

Remembering all coefficients are in mod 8 form, equation (4.29) can now be manipulated algebraically in order to render an expression that lends itself well to direct implementation. Such a form should contain to the greatest possible extent expressions that are generated as transition pulses during the addition of 1, such as $B = \beta_B$, $\bar{B} = \alpha_B$, $BA = \beta_A$, $B\bar{A} = \alpha_A$, etc. It should also be as free as possible of expressions that must be separately generated, such as $\bar{C}$, $\bar{C}B$, etc. Finally, the coefficients of the terms should contain as few 1's as possible, preferably one. This will minimize timing problems.

For the present synthesis, such a form is

$$v = 1 + 4\bar{C}BA + CBA + 3\bar{C}B\bar{A} \qquad (4.30)$$

The first three terms are very simple to implement. A partial synthesis based on these terms is shown in Figure 4.3. Feedback line (a) performs the function of adding CBA, since there is a $\alpha_{\bar{C}}$ pulse produced when 1 is added to an initial reading of 011, and its connection back to stage B corresponds to an addition of 1 in such a case.

Line (b), the set input to stage $\bar{C}$ replacing the original

trigger input, can be explained as follows: For an initial state $\bar{C}AB$, the addition of 1 will produce a carry signal β_A, which would normally trigger stage $\bar{C}$. But the term $4\bar{C}AB$ indicates that an additional 4 should be added in such a case, or in other words stage $\bar{C}$ should be triggered again, voiding the effect of the original trigger. Rather than perform this cumbersome operation, it is easier simply to avoid either transition when the initial state of $\bar{C}$ is 1. This is done simply by changing the trigger input to a set input.

There remains yet the problem of implementing the rest of the synthesis. The present circuit functions properly everywhere except, of course, when the initial state is $\bar{C}\bar{A}\bar{B}$ (i.e. 100). Here, stage $\bar{C}$ should be reset and stage B should be prevented from going to a stable state of 1 as it would do in the present circuit. The resetting of stage $\bar{C}$ is best accomplished directly at event time 0t by the signal $\bar{A}\bar{B}p$ generated by means of an AND gate and applied to the reset input. (This signal will appear also for an initial state of 000, but will have no effect since it is a reset signal). The action of stage B is best corrected by blocking the command signal p from its trigger input when the circuit is in the 100 state. Allowing a transition and correcting

subsequently is not advisable, primarily because the level signal $\bar{B}$ is being used to form the signal $\bar{A}\bar{B}p$ previously mentioned. The required signal blockage is specified by

$$\Delta_B p = \overline{(\bar{C}\bar{A}\bar{B})}p = (C + A + B)p \qquad (4.31)$$

The most economical method of implementing this requirement, componentwise, is by a T input:

$$T_B\, p = (C + A + B)\, p \qquad (4.32)$$

As a further measure toward component economy let the feedback line from stage $\bar{C}$ to stage B (labeled (a) in Figure 4.3) be moved from the trigger input of B to its set input. The effect of the line remains the same since the state of stage B is always 0 when a pulse appears on line (a), yet the need for an additional OR gate at the input to stage B is thus obviated.

The final form of the synthesized circuit is shown in Figure 4.4. Due to the liberal use of set and reset inputs, this circuit is highly stable. The reduced connection matrices are

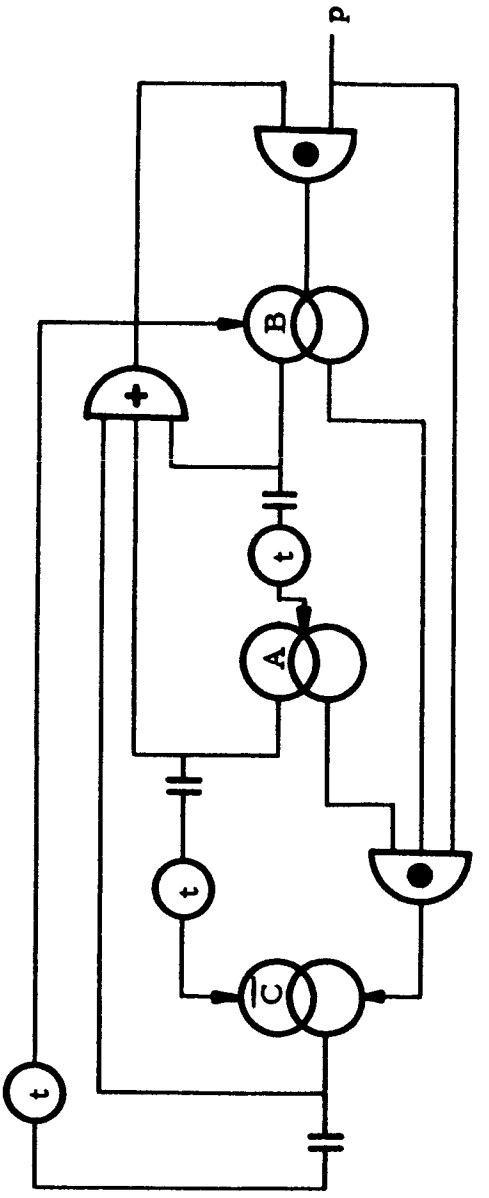


Figure 4.4

$$\left[\Pi_C\right]_r^t = \left[\Pi_C\right]_r^{2t} = \left[0\right] \tag{4.33}$$

$$\left[\Pi_C\right]_r^{3t} = \begin{matrix}\Delta_C \\ \Delta_B \\ \Delta_A\end{matrix} \begin{pmatrix} \bar{C}A^tB^{2t}\bar{C}^{3t} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

The zero rows for stages B and A indicate that these stages cannot form part of an unstable loop. Yet, the loop term $\bar{C}A^tB^{2t}\bar{C}^{3t}$ indicates they are necessary for the only possible unstable loop the circuit could describe.

Therefore, it follows that the circuit is inherently stable.

The subinterval analysis for the circuit is shown below.

				p	p^t	p^{2t}	p^{3t}	p^{4t}
Δ_C	0	0	$\bar{C}A$	$\underline{\bar{C}}\underline{A}\underline{B}$	0	$\underline{C}\underline{A}\underline{B}$	0	0
Δ_B	BC	0	0	$\underline{A}+\underline{B}+\underline{C}$	~~$BC\underline{\bar{C}}\underline{A}\underline{B}$~~	0	$\underline{C}\underline{A}\underline{B}$	0
Δ_A	0	B	0	0	$B(\underline{A}+\underline{B}+\underline{C})=B$	0	0	~~$B\underline{C}\underline{A}\underline{B}$~~

(4.34)

The logic of the circuit is given by

$$C' = C \oplus \bar{C}A\bar{B} \oplus CAB = A\bar{B} + \bar{A}C + \bar{B}C$$

$$B' = B \oplus (A+B+C) \oplus CAB = A\bar{B} + AC + \bar{B}C \quad (4.35)$$

$$A' = A \oplus B = A\bar{B} + \bar{A}B$$

The examples thus far treated have been confined to the non-redundant case. That is, the desired count sequence contained every one of the 2^n possible n-bit words. Very often, however, this is not the case. Instead, there are r redundant combinations, where $r < 2^n$, to which there correspond r equations of the form

$$m_j \ (A, B, C, ...) = 0 \qquad (4.36)$$

as was pointed out on page 119. It was also indicated that to each redundant word, any value of v may be assigned. As a result, the values of the w coefficients are no longer unique but are rather linearly dependent on a set of r arbitrary constants. To demonstrate how this case is treated, a typical synthesis problem will now be considered:

A 5211 BCD (Binary Coded Decimal) counter is to be designed. The required count sequence is shown in Table 4.5. Examining the periodic characteristics of the various stages, it is found

	A B C D	v	COMPOSITION
non-redundant words	0 0 0 0	1	w_0
	0 0 0 1	2	$w_0 + w_1$
	0 0 1 1	3	$w_0 + w_1 + w_2 + w_3$
	0 1 1 0	1	$w_0 + w_2 + w_4 + w_6$
	0 1 1 1	1	$w_0 + w_1 + w_2 + w_4 + w_3 + w_5 + w_6 + w_7$
	1 0 0 0	1	$w_0 + w_8$
	1 0 0 1	2	$w_0 + w_1 + w_8 + w_9$
	1 0 1 1	3	$w_0 + w_1 + w_2 + w_8 + w_3 + w_9 + w_{10} + w_{11}$
	1 1 1 0	1	$w_0 + w_2 + w_4 + w_8 + w_6 + w_{10} + w_{12} + w_{14}$
	1 1 1 1	1	$\sum_j w_j$
	0 0 0 0	—	— — —
redundant words	0 0 1 0	a	$w_0 + w_2$
	0 1 0 0	b	$w_0 + w_4$
	0 1 0 1	c	$w_0 + w_1 + w_4 + w_5$
	1 0 1 0	d	$w_0 + w_2 + w_8 + w_{10}$
	1 1 0 0	e	$w_0 + w_4 + w_8 + w_{12}$
	1 1 0 1	f	$w_0 + w_1 + w_4 + w_8 + w_5 + w_9 + w_{12} + w_{13}$

Table 4.5

that all positions are already in an acceptable order. Similarly, the choice of variables appears excellent, with a w_0 of 1 and the values of v indicating small positive increments throughout. The six redundant combinations are listed underneath the required sequence and symbols a through f have been chosen to represent the corresponding values of v. These values are arbitrary, with this exception: Let it be specified that if the counter ever assumes a redundant state, as a result of a spurious input, it will return to a permissible state in the next count. This error-correcting feature will be considered after a preliminary synthesis has been completed.

Table 4.6 lists the w coefficients (shown with their corresponding exterms for easier reference) obtained from Table 4.5. All but w_0, w_1, w_8, and w_9 are seen to be linearly dependent on one or more of the six arbitrary constants a through f. Clearly, the selection of the values for these entities will greatly affect the simplicity of the synthesis. Since $w_0 = 1$, the exterms that are easily available in pulse form are D, CD, BCD, and ABCD. On the other hand, quantities such as B, C, AB, etc., are difficult to obtain without gating and every attempt should be made to make their coefficients vanish. For this synthesis, the best choice of constants by far is

w Coefficient	Exterm	Value
w_0	1	1
w_8	A	0
w_4	B	b-1
w_2	C	a-1
w_1	D	1
w_{12}	AB	e-b
w_{10}	AC	d-a
w_9	AD	0
w_6	BC	2-a-b
w_5	BD	c-b-1
w_3	CD	2-a
w_{12}	ABC	a+b-e-d
w_{13}	ABD	f+b-c-e
w_{11}	ACD	a-d
w_7	BCD	a+b-c-2
w_{15}	ABCD	c+d+e-a-b-f

Table 4.6

$$\begin{aligned} a &= 1 \\ b &= 1 \\ c &= 2 \\ d &= 1 \\ e &= 1 \\ f &= 2 \end{aligned} \qquad (4.36)$$

The resulting synthesis equation is

$$v = 1 + D + CD - 2BCD \qquad (4.37)$$

This may be manipulated as follows:

$$v = 1 + D(1 + C) - 2BCD \qquad (4.38)$$

$$= 1 + D(2C+\bar{C}) - 2BCD \qquad (4.39)$$

$$= 1 + \bar{C}D + 2CD - 2BCD \qquad (4.40)$$

$$= 1 + \bar{C}D + 2CD(1-B) \qquad (4.41)$$

$$= 1 + \bar{C}D + 2\bar{B}CD \qquad (4.42)$$

Equation (4.42) is very easy to implement. In the addition of 1, $\bar{C}D$ produces a α_C pulse at event time 3t, while $\bar{B}CD$ produces a α_B pulse at 4t. To help matters, both signals cannot occur in the same count and neither signal interferes with the addition of 1. The term $\bar{C}D$ indicates α_C must be fed back to stage D to add an extra 1. Since stage D is always in the 0 state in such a case (the α_C was generated by a β_D pulse) this signal will be applied to the set input of stage D.

Similarly, the term $2\bar{B}CD$ indicates α_B must be fed back to stage C in order to add an extra 2. Again, since α_B was generated by a β_C signal, stage C must be in the 0 state, and the feedback signal may therefore be applied at the set input of stage C.

The specifications for the synthesized circuit are

$$\begin{aligned} T_D &= p \\ S_D &= \alpha_C^t \\ T_C &= \beta_D^t \\ S_C &= \alpha_B^t \\ T_B &= \beta_C^t \\ T_A &= \beta_B^t \end{aligned} \qquad (4.43)$$

As in the previous example, the circuit is highly stable due to the use of set inputs in the feedback loops. The reduced connection matrices are

$$\left[\Pi_C\right]_r^t = \left[\Pi_C\right]_r^{3t} = \left[0\right] \qquad (4.44)$$

$$\left[\Pi_C\right]_r^{2t} = \begin{array}{c} \\ \Delta_D \\ \Delta_C \\ \Delta_B \\ \Delta_A \end{array} \begin{array}{c} \begin{array}{cccc} \quad & \quad & \quad & \quad \end{array} \\ \begin{pmatrix} \bar{D}\bar{C}^t{}_D{}^{2t} & 0 & 0 & 0 \\ 0 & \bar{C}\bar{B}^t{}_C{}^{2t} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \qquad (4.45)$$

$$[\Pi_C]_r^{4t} = \begin{array}{c} \\ \Delta_D \\ \Delta_C \\ \Delta_B \\ \Delta_A \end{array} \begin{pmatrix} \bar{D}\bar{C}^t\bar{B}^{2t}C^{3t}D^{4t} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{4.46}$$

The empty Δ_A and Δ_B rows indicate that neither stage A nor B can be part of an unstable loop. This makes C stable since its only possible loop depends on B. The stability of C, in turn, establishes that of D for the same reason.

The synthesized counter is shown in Figure 4.5. The subinterval analysis appears below

$$\begin{array}{c} \\ \Delta_D \\ \Delta_C \\ \Delta_B \\ \Delta_A \end{array} \begin{pmatrix} 0 & D\bar{C} & 0 & 0 \\ D & 0 & C\bar{B} & 0 \\ 0 & C & 0 & 0 \\ 0 & 0 & B & 0 \end{pmatrix} \qquad \begin{array}{c|c|c|c|c} p & p^t & p^{2t} & p^{3t} & p^{4t} \\ \hline 1 & 0 & \underline{\bar{C}D} & 0 & \cancel{D\bar{C}\bar{B}CD} \\ 0 & \underline{D} & 0 & \cancel{D\bar{C}D}+\underline{\bar{B}CD} & 0 \\ 0 & 0 & \underline{CD} & 0 & \cancel{C\bar{B}CD} \\ 0 & 0 & 0 & \underline{BCD} & 0 \end{array} \tag{4.47}$$

At this point, the problem of error correction may be taken up. From the subinterval analysis in (4.47) it may be seen that redundant states lead to the following transitions in response to p:

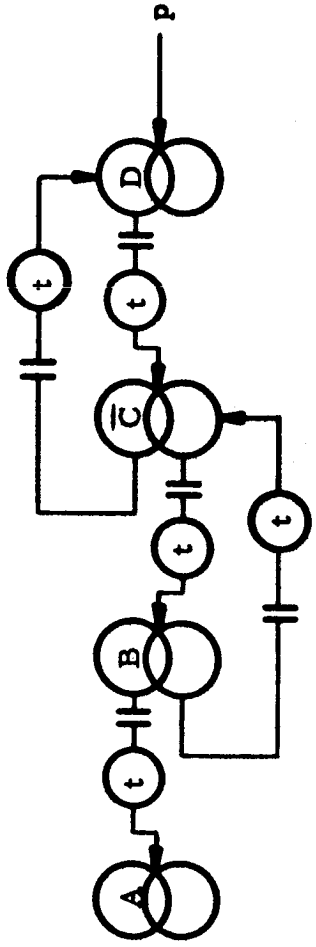


Figure 4.5

Redundant state	Next state	Correction OK?
0 0 1 0	0 0 1 1	Yes
0 1 0 0	0 1 0 1	No
0 1 0 1	0 1 1 1	Yes
1 0 1 0	1 0 1 1	Yes
1 1 0 0	1 1 0 1	No
1 1 0 1	1 1 1 1	Yes

It is apparent that the circuit's present ability to correct errors is satisfactory except for the redundant words (0100) and (1100). For these words, one more count is required to bring the counter into a permitted state. Therefore, additional correction must be provided for these two cases. The simplest way of doing this is to generate an an additional command pulse when these two words are present.

The (0100)→(0101) and (1100)→(1101) transitions are characterized by $B\bar{C}\alpha_D$. This can be easily obtained as a pulse by means of an AND gate and fed back to D for an additional command pulse. Since D is always in the 1 state when this signal is applied, a reset input may be used. The revised circuit is shown in Figure 4.6. It remains highly stable, like its simpler version in Figure 4.5.

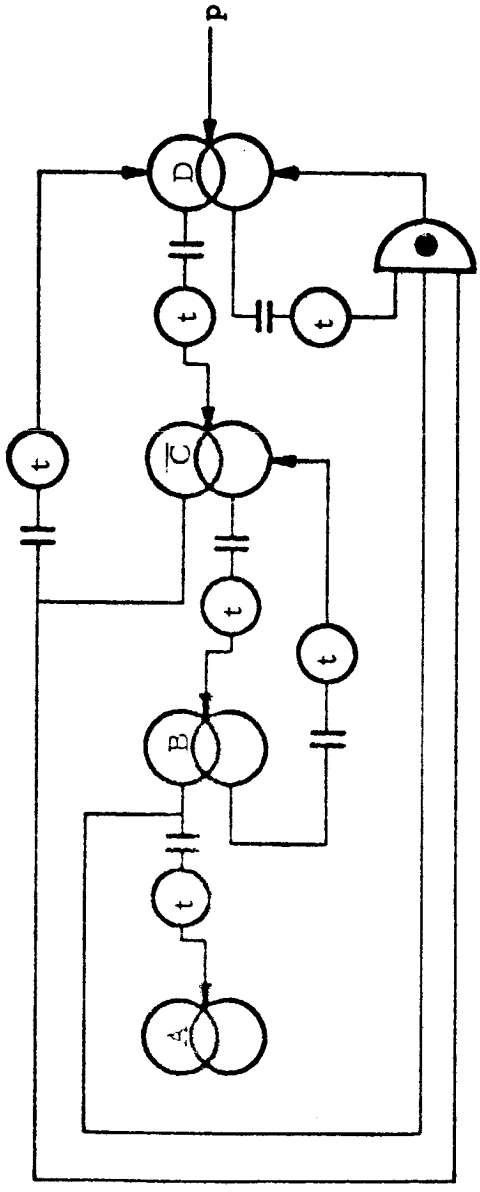


Figure 4.6

Whether or not the one-count error correction feature is worth the added cost for the circuit is a matter to be decided by the performance requirements. Clearly, if spurious inputs occur rarely enough to make occasional two-count error correction tolerable, the very economical, gateless circuit shown in Figure 4.5 is eminently satisfactory.

The synthesis technique presented in this chapter is predicated on the contention that a standard binary adder can be modified to count in any other code within its capacity. The ease with which this adaptation is made depends on the character of the code in question. Regardless of the code, however, the synthesis equation represented by the weighted exterm expansion for v is rarely in a form that is directly implementable. Almost without exception, such a direct implementation would be so costly in terms of logic and so fraught with timing problems that no justification could exist for its use even from the standpoint of mathematical elegance. Yet, as was seen in the examples presented in this chapter, a well-chosen algebraic manipulation of the basic weighted-exterm synthesis equation may yield a form that points clearly and unequivocally toward some simple method of implementation by transition coupling. As to how such forms are generated, no clear-cut strategy is known for this

purpose. The mode of attack exemplified by Equations 4.36 - 4.42 is based entirely on heuristic reasoning and generally improves with experience, as easily implementable forms become increasingly easier to recognize. As a rule, if a sizable portion of the synthesis equation appears in a form that is easily and economically implementable, such an implementation should be made and a superposition of less economical corrections for the remainder of the synthesis should be attempted.

V. CONCLUSIONS AND RECOMMENDATIONS

The guiding purpose of this study has been the development of a logic rationale whose structure would be capable of accomodating the transition-coupled asynchronous counter. Its practical role would be to place the analysis and synthesis of this often-used device on a formal foundation suitable for exploitation by the engineer.

The emphasis in this paper on the counter rather than on its multi-input generalization constitutes a restriction of a motivational, rather than conceptual, nature: With the exception of certain simple arithmetic units, transition-coupled multi-input circuits are virtually unheard of today, primarily because of extremely poor reliability.

Nevertheless, the mathematical concepts of subinterval logic and the practical techniques associated with it are capable of being extended to the general multi-input case. In anticipation of a time when engineering objectives will warrant such extension, some of the basic groundwork for it is here presented in way of a recommendation for further study:

As was seen in Chapter III, the action of a transition-coupled counter is fully determined by:

a. The connection matrix $[C]^t$,

b. The starting state, and

c. The zeroth transition vector $[T_0]$

However, the pulse input vector at time 0t determines only the zeroth transition vector. It has no effect whatsoever on $[C]^t$ and the question of how a circuit came to be in a particular starting state is certainly irrelevant to the analysis.

Now, the circuit analysis algorithm developed in Chapter III yields a timing chart of transitions by every stage as a function of the starting state, for one particular input vector characterized by the zeroth vector $[T_0]$. Such a chart thus furnishes a complete description of the switching action of a counter.

It is now but a brief step to extrapolate this reasoning to the general case. A different input vector generates a different zeroth vector $[T_0]$ and corresponding timing chart. The various charts could then be combined in the following basically equivalent ways:

a. Transition matrix method: Instead of computing a transition vector at each event time, kt, a n-column transition matrix $[\mathcal{T}]^{kt}$ is computed, each

of whose columns is a transition vector corresponding to a particular input. Being simply a superposition of n timing charts, the method tends to become notationally cumbersome as n increases.

b. Direct superposition method: In this method, the transition vector is expressed as the union of transition vectors associated with a set of independent inputs. For example, letting a_0 and a_1 be two independent inputs, (e.g. addend digits to a binary adder) the k^{th} transition vector can then be given by

$$\left[T_k\right] = \bar{a}_0\bar{a}_1 \left[T_k(0,0)\right] + \bar{a}_0 a_1 \left[T_k(0,1)\right] + a_0\bar{a}_1 \left[T_k(1,0)\right] + a_0 a_1 \left[T_k(1,1)\right] \tag{5.1}$$

As an illustration, the subinterval analysis for a 3-bit binary adder with input lines a_0 and a_1 -- for the purpose of addition of $2a_1 + a_0$ to the contents of the adder -- is shown on the next page:

		p	p^t	p^{2t}	p^{3t}
Δ_C	0 0 0	$\underline{a}_0$	0	0	0
Δ_B	C 0 0	$\underline{a}_1$	$\underline{C}\underline{a}_0$	0	0
Δ_A	0 B 0	0	$\underline{B}\underline{a}_1$	$\underline{B}\underline{C}\underline{a}_0=(\underline{B} \oplus \underline{a}_1)\underline{C}\underline{a}_0$	0

(5.2)

As may be seen, any vector in (5.2) can be easily expressed in the form of (5.1) since its general form is available. Unfortunately, obtaining a general form is usually not as simple as it has been in this particularly tractable case. The principal difficulty lies, of course, in referring each non-underlined literal to event time Ot. The underlined form of this literal will now depend not only on the starting state of the circuit but also on the values of the input variables -- most often in increasingly complex relationships. For example, $(\underline{B} \oplus \underline{a}_1)\underline{a}_0\underline{C}$ shows such a dependency. Were the circuit somewhat more complicated, and the term $\underline{A}\underline{B}\underline{C}$, say, appeared somewhere in the succeeding transition vector, then the underlined counterpart of A would depend on a_0 and a_1, according to whose values $(\underline{B} \oplus \underline{a}_1)\underline{a}_0\underline{C}$ can be equal to 0, BC, or $\bar{B}C$. The general expression for $\underline{A}\underline{B}\underline{C}$ would be $(\underline{\bar{A}} \oplus \underline{a}_0 \oplus \underline{a}_0\underline{\bar{a}}_1)\underline{B}\underline{C} = \left[\underline{A} \oplus (\underline{a}_0 + \underline{a}_1)\right] \underline{B}\underline{C}$. This, in

turn, would generate an even more complicated expression in some succeeding transition vector, and so forth.

In brief, what is seriously needed is a new, sophisticated form of "shorthand" notation, analogous to that used in the counter analysis technique, which would be capable of rendering a compact yet concise description of the action of a multi-input transition-coupled circuit.

It might also be noted that the stability question presents no new problem in the multiple-input case, for as was indicated in Chapter II, inherent stability is a characteristic of the connection matrix, and is independent of external inputs, assuming, of course, that the inputs are themselves stable.

Naturally, stability remains a major engineering problem in its own right, even for the single-input case, simply because of the tedious process involved in ascertaining it. Formulation of a simple general stability test for transition-coupled circuits, therefore, constitutes one of the principal topics in which further research is recommended.

Another objective toward which further study should be directed is that of broadening the mathematical model of the transition-coupled circuit and correspondingly extending

analysis techniques where needed. The mathematical concepts of subinterval logic, introduced in Chapter II can be seen to envision a limited type of transition coupling. Specifically, the interstage delays are very nearly the same throughout the circuit and are much larger than those associated with the combinational logic components in the circuit. This model corresponds well with the majority of the transition-coupled circuitry in use today.

The peculiar character of the logic subinterval -- the dominant concept of Chapter II -- does lead to one important reservation from the mathematical as well as practical standpoint, arising from the nature of the asynchronism embodied in the time increment $\delta_k \ll t$ during which signals generated in the past arrive at their destinations. In its present formulation, subinterval logic requires that δ_k remain very small compared to t throughout the switching action of the circuit between successive stable states. Only in this way can every memory transition be unambiguously related to an "event time." Yet it is a fact that this quantity will tend to grow with the event time. Namely, recalling that t has been specified as the longest interstage delay in the circuit, the only general statement regarding δ_k that can be made is

$$0 \leq \delta_k \leq k\delta_1 \qquad (5.3)$$

This follows because the earliest signal arrival associated with event time kt would be the result of the shortest delay in the circuit repeated k times. This is given by $k(t-\delta_1)$. The latest first arrival would be at time kt itself, corresponding to a k-fold repetition of the longest delay in the circuit.

Since δ_k is the difference between kt and the time of the first arrival, it is seen that the range of δ_k is described by equation (5.1).

It must be concluded, then, that there is an implicit limitation on the structure of the circuits treated by subinterval logic. For should the earliest pulse signal arrival for an event time occur before all level signals associated with this event time have ahd time to stabilize, an incorrect result will be reached. To determine the simplest requirement for prevention of this critical race condition, let τ_{max} represent the longest level signal rise time in the circuit. This may be the sum of the rise times of a memory unit and a cascade of gates, or simply the rise time of the slowest memory unit.

Now consider an event associated with event time kt. Stability of all level signals associated with this time can be guaranteed no sooner than $(k-1)t + \tau_{max}$. The earliest pulse signal associated with time kt, arriving as early as $kt-k\delta_1$, must find the level signals in the circuit stable. As a result, the simplest requirement for proper operation of the circuit at time kt (providing it operated properly up to this time) is

$$(kt - k\delta_1) > (kt - t + \tau_{max}) \tag{5.4}$$

or

$$\delta_1 < \frac{t - \tau_{max}}{k} \tag{5.5}$$

Finally, since the circuit must function properly for any transition, equation (5.5) becomes

$$\delta_1 < \frac{t - \tau_{max}}{M_{max}} \tag{5.6}$$

where M_{max} is the largest number of event times (not including Ot) the counter in question requires for transition between two stable states.

What this requirement means is simply that the larger the rise times in the circuit and the number of event times required by the circuit to go between stable states, the

smaller must be the quantity δ_1, i.e. the more nearly equal must be the interstage delays.

The requirement represented by inequality (5.4) has been assumed satisfied throughout the preceding study. Nevertheless, it should be realized that in some cases, this inequality may conceivably not be easy to satisfy. Picture, for example, a circuit with a complex feedback structure and a correspondingly large M_{max} (e.g., the circuit in Figure 2.6, with an M_{max} of 6). If the circuit contained in addition relatively long rise times, the δ_1 required for reliable operation according to the subinterval logic model might be so small as to introduce a new parameter into the design of the interstage combinational circuitry. Specifically, the objective of the design would now become the provision of the required uniformity in the interstage delays, in addition to the minimization of logic circuitry or component cost.

Circuit synthesis with uniform interstage delay time as a design parameter was not considered in this paper. It is recommended as a topic for further study.

Another area in which additional investigation is suggested is an extension of the subinterval logic philosophy to include circuits with different interstage delays. The problem is

meaningful not only because different types of memory units could be used in the same circuit but also because high switching speeds tend to make delays encountered in gating more and more significant in relation to the interstage delay. Rather than forcibly providing a uniform interstage delay, designs can be developed which take advantage of the non-uniformity and the subinterval analysis technique can be extended to accommodate it. What is required, basically, is a subinterval length t compatible with all delays and a consistent notation to identify the number of event times required for various actions in the circuit, as well as the availability of a logical quantity as a signal. (For example, a pulse applied at event time kt to a gate with a delay of 2t cannot be expected at the output of the gate until event time (k+2)t.)

Perhaps the most challenging area for further research, however, suggests itself in Chapter IV. In that chapter, the binary adder was chosen as a "model" for the synthesis of transition-coupled counters simply because it happens to be a multi-input transition coupled circuit whose design is well known, whose simplicity, reliability, and freedom from timing problems are unquestionable, and whose action is easy to describe in an extremely compact notation, that of standard algebra.

Yet, it is a fact that the standard binary adder is not an ideal model for all types of counters. For example, a general two-decade BCD counter would be far simpler to design as a transition-coupled unit if an 8421 BCD adder, say, were used as a model than if the standard binary adder were employed.

A significant contribution could therefore be made if a convenient set of models were developed (perhaps by using the binary adder as a preliminary model) for the purpose of circuit synthesis. For usefulness, these models would have to be free of timing problems, their physical and logical properties would have to be thoroughly analyzed, and a consistent and convenient notation should be available to describe their action. Hopefully, the outgrowth of this development would be algorithms to determine the best model for a synthesis, the optimum arrangement of digit positions and the best choice of variable assignments for the new circuit.

A contribution with considerably farther-reaching implications could, in turn, evolve from the above developments: A unified model adaptation algorithm, by means of which <u>any</u> circuit already designed could readily play the role of model for one to be synthesized.

Summary

A logic capable of describing the transition-coupled asynchronous counter with approximately uniform interstage delays was developed in Chapter II. It has been named subinterval logic. A method of ascertaining the stability of a transition-coupled counter, using subinterval logic, was also described in Chapter II. A simple subinterval logic technique for the analysis of a stable transition-coupled counter was derived in Chapter III. A standard-algebraic method for synthesizing transition-coupled counters was described in Chapter IV. Recommendations for further study were presented in Chapter V.

12. Ginsburg, S. "A Technique for the Reduction of a Given Machine to a Minimal State Machine," IRE Transactions on Electronic Computers, Volume EC-8, September, 1959, pp. 346-356.

13. Ginsburg, S. "Synthesis of Minimal State Machines," IRE Transactions on Electronic Computers, Volume EC-8, December 1959, pp. 441-449.

14. Hartmanis, J. "On the State Assignment Problem for Sequen- Machines I," IRE Transactions on Electronic Computers, Volume EC-10, June 1961, pp. 157-165.

15. Hohn, F. E., Seshu, S., and Aufenkamp, D. D. "The Theory of Nets," IRE Transactions on Electronic Computers, Volume EC-6, September 1957, pp. 154-161.

16. Huffman, D. A. "Synthesis of Sequential Switching Circuits," Journal of the Franklin Institute, Volume 257, No. 3, March 1954, pp. 161-190; No. 4, April 1954, pp.275-303.

17. Huffman, D. A., Study of the Memory Requirements of Sequential Switching Circuits, Technical Report 293, Electronics Research Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, April 1955.

18. Huffman, D. A. "Problems of Sequential Circuit Synthesis," Session on Advanced Theory of Logical Design of Digital Computers, University of Michigan, Ann Arbor, Michigan, June 1958.

19. Kleene, S. C. "Representation of Events in Nerve Nets and Finite Automata," Automata Studies, Annals of Mathematical Studies, C. E. Shannon and J. McCarthy, Editors. Princeton University Press, Princeton, New Jersey, 1956, pp. 3-41.

20. Maley, G. A., and Earle, J. The Logic Design of Transistor Digital Computers, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1963.

21. Marcus, M. M. Switching Circuits for Engineers, Prentice Hall, Inc. Englewood Cliffs, New Jersey, 1962.

22. McCluskey, E. J., Jr. and Unger, S. H. "A Note on the Number of Internal Variable Assignments for Sequential Switching Circuits," IRE Transactions on Electronic Computers, Volume EC-8, December 1959, pp. 439-440.

23. Mealy, G. H "A Method for Synthesizing Sequential Circuits," Bell System Technical Journal, Volume 34, No. 5, September 1955, pp. 1045-1079.

24. Mergler, H. W. "Counting and Scaling," Digital Systems Engineering, Volume I, H. W. Mergler, Editor. Case Institute of Technology, Cleveland, Ohio, June 1961, Chapter 4.

25. Metze, G., Miller, R. E. and Seshu, S "Transition Matrices of Sequential Machines," IRE Transactions on Circuit Theory, Volume CT-6, March 1959, pp. 5-11.

26. Moore, E. F. "Gedanken-Experiments on Sequential Machines," Automata Studies, Annals of Mathematics, C. E. Shannon, J. McCarthy, Editors, Princeton University Press, Princeton, New Jersey, 1956, pp. 129-153.

27. Muller, D. E. "Asynchronous Switching Theory," Session on Advanced Theory of Logic Design of Digital Computers, University of Michigan, Ann Arbor, Michigan, June 1958.

28. Netherwood, D. B. "Minimal Sequential Machines," IRE Transactions on Electronic Computers, Volume EC-8, September 1959, pp. 367-380.

29. Radchenko, A. N. and Filippov, V. I. "Shift Registers with Logical Feedback and Their Use as Counting and Coding Devices," Automation and Remote Control, Volume 20, No. 11, November 1959.

30. Stearns, R. E. and Hartmanis, J. "On the State Assignment Problem for Sequential Machines, II," IRE Transactions on Electronic Computers, Volume EC-10, December 1961, pp. 593-603.

31. Unger, S. H. "Simplification of State Tables," A Survey of Switching Circuit Theory. E. J. McCluskey, Jr. and T. C. Bartee, Editors, Mc Graw Hill Book Co., New York, 1962, Chapter 9.

32. Unger, S. H. and Paull, M. C. "Minimizing the Number of States in Incompletely Specified Sequential Switching Functions," IRE Transactions on Electronic Computers, Volume EC-8, September 1959, pp. 356-366.

33. Unger, S. H. "Hazards and Delays in Asynchronous Sequential Switching Circuits," IRE Transactions on Circuit Theory, Volume CT-6, March 1959, pp. 12-25.

34. Unger, S. H. A Study of Asynchronous Logical Feedback Networks, Technical Report No. 320, Electronics Research Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1957.